MASTER EN SCIENCES INFORMATIQUES

Gestion des connaissances: développement d'un prototype

Cossement, Alexis; Rochet, Jean-Philippe

Award date:
2001

Facultés Universitaires Notre-Dame de la Paix,
Institut d'Informatique
Namur

Année académique 2000-2001

Gestion des connaissances :
Développement d'un prototype

Alexis Cossement
Jean-Philippe Rochet

Mémoire présenté en vue de l'obtention
du grade de Maître en Informatique

Résumé : La gestion des connaissances est un domaine encore peu formalisé au sein des diverses organisations même si elle est tout doucement en expansion. Développée au travers de démarches pragmatiques, elle recouvre aujourd'hui un vaste ensemble de méthodes bien structurées ainsi que des outils adaptés à ces méthodes. Il est important de souligner l'intérêt, les objectifs mais aussi les difficultés et les limites d'une gestion des connaissances au sein d'une organisation. C'est pour répondre à ces objectifs et contourner ces limites que nous avons contribué au développement d'un prototype de gestion des connaissances dénommé Anthémis. Ce prototype suit deux approches de développement. La première approche, selon les modèles Entité-Association, est une approche beaucoup plus classique et pragmatique et a pour but la création d'un logiciel de gestion des connaissances innovant. L'approche selon les graphes conceptuels est, quant à elle, beaucoup plus expérimentale et a pour but premier la validation de la puissance du formalisme des graphes conceptuels pour la gestion des connaissances ainsi que la comparaison de ce formalisme avec le formalisme des modèles Entité-Association. Le prototype Anthémis offre la possibilité aux utilisateurs de créer leurs propres modèles, modèles qui définiront la structure des connaissances à intégrer par après. Ainsi l'utilisateur n'est pas limité à un modèle prédéfini par les créateurs du logiciel. De plus l'insertion des modèles et des connaissances ne nécessitera dans l'avenir aucun pré-requis pour l'utilisateur. Le prototype réalise aussi une présentation des connaissances de manière standardisée et uniforme malgré la variété des modèles qui peuvent être créés. Pour finir, le prototype a montré, pour les graphes conceptuels, sa capacité à gérer des connaissances et sa puissance par rapport aux modèles Entité-Association par l'utilisation entre autre d'un moteur de recherche évolué.

Abstract : Today knowledge management is not much formalized within the different organisations eventhough it is slowly expanding. Developed through pragmatic approaches, it comprises a large scope of well-structured methods as well as tools adapted to these. It is important to emphasize not only the utility, the objectives but also the difficulties and limits of knowlege management within an organisation. In order to meet these objectives and to go beyond these limits we have contributed to the development of a prototype called Anthémis to manage knowledge. We have adopted two different approaches. Firstly a more classical and pragmatic approach according to the Entity-Association models in order to create an innovative software to manage knowledge. Secondly a more experimental approach has been developed according to the conceptual graphs. The main objective of this approach is to validate the power of the formalism of conceptual graphs to manage knowledge and to compare this formalism with the formalism of the Entity-Association models. The prototype Anthémis gives the users the possibility to create their own models, which will define the structure of knowledge to introduce afterwards. In this case the user is not limited by a model defined by those who created the software. Besides in the future the user will be able to introduce models and knowledge without any basic knowledge. Moreover the prototype offers a standardized and uniform presentation of knowledge despite the variety of models that can be created. At last this prototype has shown, through the conceptual graphs, its capacity to manage knowledge and its power in comparison with the Entity-Association models, among other through the use of an advanced searching device.

Remerciements

Nous tenons à remercier spécialement notre promotrice, madame Claire Lobet-Maris, pour son soutien durant notre travail et pour son suivi lors de notre rédaction ainsi que monsieur Olivier Gerbé pour son accueil et son encadrement durant le développement du prototype Anthémis. Nous désirons vivement les remercier de nous avoir offert la possibilité de réaliser ce travail.

Nous voulons remercier notre copromotrice, mademoiselle Anne Devos, pour ses conseils, son suivi et son expérience dans le domaine.

Nous remercions également madame Béatrice Van Bastelaer et mademoiselle Christine Marsigny pour leurs conseils et leur écoute lors de nos réunions de travail ainsi que messieurs Jean Henrard et Radu Cotet pour leur aide lors de la préparation de la démo de notre prototype.

Nous tenons finalement à remercier mademoiselle Hélène Lefèvre pour tout le temps qu'elle a consacré lors de la correction de notre présent travail ainsi que toutes nos familles et tous nos amis qui nous ont soutenus durant toutes ces années.

Table des matières

Introduction

La gestion des connaissances est à l'heure actuelle une force encore non exploitée par un grand nombre d'organisations. Elle est, comme nous le verrons, un élément moteur pour le bon fonctionnement d'une organisation même si sa mise en œuvre reste encore difficile. Ne vaut-il pas mieux en effet une seule connaissance collective que de multiples connaissances individuelles ? C'est pour répondre à cette question que nous allons dans ce présent travail réaliser une description de la gestion des connaissances ainsi que du prototype de gestion des connaissances que nous avons développé. Nous verrons ainsi que la gestion des connaissances est en pleine expansion. Nous présenterons, dès lors, plusieurs logiciels et méthodes de gestion des connaissances qui existent sur le marché ainsi que le prototype Anthémis dont nous avons collaboré à la réalisation dans le cadre de notre stage de fin d'année.

Notre travail sera divisé en deux parties, l'une présentant les pré-requis théoriques de la gestion des connaissances et l'autre concernant la création d'un prototype pour cette gestion. Nous commencerons ainsi à présenter de manière générale la gestion des connaissances avant de rentrer dans le vif du sujet avec la présentation du prototype Anthémis.

Dans la première partie, nous introduirons dans un premier chapitre la gestion des connaissances. Nous détaillerons ainsi, les notions en rapport avec cette gestion, les motivations qui ont amené l'engouement pour celle-ci, ses objectifs ainsi que ses avantages et ses limites.

Une fois ces différents points assimilés, nous pourrons aborder dans le deuxième chapitre différentes méthodes de gestion des connaissances. Ce chapitre nous permettra ainsi de comprendre son fonctionnement. Nous ne verrons pas toutes les méthodes existantes. Nous nous limiterons à celles qui sont le plus souvent reprises par la littérature. Nous passerons ainsi en revue les méthodes MACAO (Méthode d'Acquisition des Connaissances Assistée par Ordinateur), KADS (Knowledge Analysis and Design System), REX (Retour d'EXpérience) et MKSM (Method for Knowledge System Management).

Dans la seconde partie concernant la création du prototype Anthémis, nous débuterons par le troisième chapitre qui présentera les généralités de ce prototype. Nous y détaillerons ainsi les pré-requis nécessaires pour l'élaboration du prototype ainsi que les objectifs que celui-ci poursuit. Nous verrons également les deux approches suivies pour sa création, l'une se basant sur les modèles Entité-Association, l'autre concernant les graphes conceptuels. Nous aborderons alors ses différentes fonctionnalités.

Une fois les généralités du prototype assimilées, nous détaillerons dans les deux chapitres suivants l'ensemble des technologies et des langages utilisés pour la création du prototype Anthémis. Nous aborderons ces technologies et ces langages dans ce présent travail car nous ne les avions jamais étudiés ni utilisés auparavant.

Nous débuterons par les différentes technologies mises en œuvre. Ces technologies sont les bases de données ORACLE, le serveur WEB Personal Web Server, le standard ASP et le protocole ODBC. Ensuite, nous parlerons des différents langages utilisés dans lesquels nous retrouvons les langages HTML, XML, XSL, Jscript ainsi que l'interface ADO.

Nous verrons alors dans le sixième et le septième chapitre la création du prototype suivant l'approche des modèles Entité-Association et celle des graphes conceptuels.

Nous commencerons par détailler l'approche suivant les modèles Entité-Association dans le sixième chapitre car celle-ci est la plus classique et la plus avancée. Nous détaillerons rapidement ainsi le mécanisme des bases de données relationnelles et de ces modèles avant de rentrer concrètement dans le fonctionnement du prototype suivant cette approche. Nous verrons alors l'architecture de la base de données, le mécanisme d'introduction des connaissances dans cette base de données, le mécanisme des fichiers XSL ainsi que les évolutions possibles pour cette approche.

Dans le septième chapitre nous analyserons en profondeur les graphes conceptuels en nous basant sur la thèse du Professeur Gerbé intitulée *'Un modèle uniforme pour la modélisation et la métamodélisation d'une mémoire d'entreprise'*. [GERBE00-1] Nous introduirons ainsi les graphes conceptuels. Ensuite, nous comparerons deux autres formalismes avec les graphes conceptuels pour montrer la puissance de ces derniers. Nous

présenterons le métamodèle utilisé pour représenter des connaissances de manière uniforme. Pour finir, nous détaillerons la création du prototype Anthémis en suivant le formalisme des graphes conceptuels.

Nous terminerons cette deuxième partie par le huitième chapitre qui clôturera notre travail et qui donnera une évaluation du prototype Anthémis. En effet, nous présenterons dans ce chapitre les avantages et les faiblesses du prototype. Nous réaliserons également une comparaison entre les deux approches suivies par le prototype ainsi qu'une comparaison entre celui-ci et les différentes méthodes vues dans le deuxième chapitre. Nous présenterons finalement des évolutions possibles pour le prototype Anthémis.

Il faut noter que ce mémoire est accompagné d'annexes présentant le code et les spécifications des différentes fonctions. Ces annexes ne sont pas mises dans le domaine public pour des questions de confidentialité.

Première partie

Pré-requis théoriques sur la gestion des connaissances

Premier chapitre : Introduction générale à la gestion des connaissances

Deuxième chapitre : Méthodologie concernant la gestion des connaissances

Premier chapitre

Introduction générale à la gestion des connaissances

I. Pré-requis nécessaires à la compréhension du concept de gestion des connaissances

1. Les définitions
   1.1. La donnée
   1.2. L'information
   1.3. La connaissance (le savoir)
2. Le concept de gestion des connaissances

II. Raisons de l'engouement pour la gestion des connaissances

1. La mondialisation
2. Le turnover des employés
3. La perte de temps

III. Buts de la gestion des connaissances

1. Les différents buts
2. Difficultés pour atteindre ces buts

IV. Obstacles et avantages de la gestion des connaissances

1. Les obstacles
2. Les avantages

Introduction de chapitre

Nous allons dans ce premier chapitre essayer de comprendre les diverses interprétations du concept de gestion des connaissances que donnent de nos jours certains auteurs. Nous préciserons également ce que nous entendons par ce concept. Pour ce faire, nous allons répondre à plusieurs questions qui peuvent implicitement nous venir à l'esprit. Qui de nos jours est intéressé par la gestion des connaissances ? Que cache ce concept de gestion des connaissances ? Pourquoi parler de connaissance et non d'information ? Quelles sont les différences entre connaissance, savoir, donnée et information ? En quoi les connaissances tacites diffèrent-elles des connaissances explicites ?

Une fois que nous aurons fixé nos idées sur tous ces différents concepts, nous parlerons des diverses raisons qui ont conduit les organisations de tout genre à se préoccuper de la gestion des connaissances. Nous parlerons ensuite des buts recherchés dans la gestion des connaissances ainsi que des obstacles et des avantages liés à celle-ci.

Concrètement, ce chapitre comportera quatre grandes parties. La première nous expliquera les nuances à apporter entre les différents concepts que sont la connaissance, le savoir, la donnée et l'information. Ensuite, viendra une seconde partie qui nous précisera le contexte dans lequel les organisations ont dû réagir en tenant compte de la gestion des connaissances. Une troisième partie nous expliquera les buts poursuivis par la gestion des connaissances. Une quatrième partie s'attachera à décrire les obstacles mais aussi les avantages liés à la gestion des connaissances. Finalement, nous terminerons par une conclusion de chapitre qui nous rappellera toutes les notions nécessaires à la suite de ce travail.

I. Pré-requis nécessaires à la compréhension du concept de gestion des connaissances

Depuis bien longtemps, le concept de connaissance est sujet à discussions et à analyses. Il reste un objet d'étude important pour les philosophes qui cherchent à comprendre comment définir précisément ce concept tout en essayant de trouver les moyens pour l'atteindre. La notion de connaissance est aussi devenue un sujet très intéressant pour les économistes. En effet, ceux-ci cherchent à exploiter les connaissances au sein des organisations car celles-ci sont perçues à l'heure actuelle comme faisant partie des actifs immatériels. Les connaissances sont considérées comme la richesse de l'organisation bien plus que le capital physique ou financier. [SVEIBI00] Les informaticiens, quant à eux, essayent de plus en plus de trouver des outils pour mieux gérer les connaissances. Nous présenterons ainsi dans la deuxième partie de ce travail l'outil de gestion des connaissances que nous avons contribué à créer.

1. Les définitions

Avant d'aborder en détail le fonctionnement de la gestion des connaissances, nous allons nuancer les concepts de donnée, d'information, de connaissance et de savoir. A ce stade, il nous faut signaler que la littérature ne s'est pas encore mise en accord sur l'utilité, dans le cadre de la gestion des connaissances, de distinguer les connaissances des autres concepts comme l'information, la donnée ou le savoir. Certains auteurs réalisent cette distinction ; c'est le cas de Prax. D'autres n'en voient pas l'intérêt. Nous avons décidé de suivre la vision de Prax en nuançant tous ces concepts car, comme lui, nous avons une vision opérationnelle de la gestion des connaissances. Il faut aussi noter qu'il existe plusieurs définitions différentes de ces concepts. Nous avons donc choisi les définitions qui correspondaient le mieux à nos objectifs finaux, à savoir la création d'un prototype de gestion des connaissances.

1.1. La donnée

Prax affirme qu' « une donnée est un fait discret et objectif ; elle résulte d'une acquisition, d'une mesure effectuée par un instrument naturel construit par l'homme. Elle peut être qualitative (le ciel est bleu) ou quantitative (la température extérieure est de 20°). Il n'y a normalement pas d'intention ni de projet dans la donnée, c'est ce qui lui confère son caractère d'objectivité. » [PRAX00, p35]

Des données sont donc des faits qui n'ont pas de liens entre eux.

1.2. L'information

Prax nous dit qu' « une information est une collection de données organisées pour donner forme à un message, le plus souvent sous une forme visible, imagée, écrite ou orale. La façon d'organiser les données résulte d'une intention de l'émetteur, et est donc parfaitement subjective (...). L'émetteur est certes obligatoire, car il faut qu'il y ait intention de délivrer un message à un récepteur ; en revanche ce dernier n'est pas intrinsèquement obligatoire (...). Dans le cas où il y ait un récepteur, voire plusieurs, c'est fondamentalement lui qui décidera si le message qu'il reçoit représente une information. Cela suppose qu'il possède la connaissance (ou les clés de décodage) lui permettant de transformer le message en information. » [PRAX00, p36]

Une information est donc un ensemble de données reliées entre elles. Il est important de noter ici que l'information ne fait que nourrir la connaissance. Cela veut dire que l'information, aussi rare ou précieuse soit-elle, ne suffit pas à améliorer les performances d'un travail. Pour être intéressante, il faut qu'elle soit interprétée et mise en relation avec le travail exécuté.

1.3. La connaissance (le savoir)

Dans ce présent travail, nous ne ferons pas de distinction entre la connaissance et le savoir. En effet, de nombreuses entreprises ne font pas, elles-mêmes, cette distinction car elles considèrent que les connaissances constituent une base essentielle pour le savoir. Nous regrouperons donc les concepts de connaissance et de savoir sous le concept de connaissance.

Lymayem retient comme définition de la connaissance « (...) l'ensemble des informations traitées et auxquelles on attribue un sens en les représentant et en les interprétant. Ces connaissances conditionnent les prises de décision des individus et leur réponse à un évènement particulier. » [LYMAYEM99, p2]

Nous pouvons donc dire que la connaissance est l'ensemble des informations qui ont été intégrées dans un contexte de travail.

Maintenant que nous savons ce que signifie la connaissance, nous allons organiser celle-ci en deux blocs : les connaissances explicites et les connaissances tacites [POLANYI66]. La littérature pour la distinction des connaissances tacites et explicites se retrouve, ici, en accord alors qu'elle ne l'était pas pour la différence entre la connaissance et l'information, la donnée, le savoir. Les connaissances explicites, telles que les procédures, les méthodes de travail, les formations internes, les documents, etc., peuvent facilement être traitées par un ordinateur, transmises électroniquement ou stockées dans des bases de données. Par contre, la nature subjective et intuitive des connaissances tacites, telles que l'expertise et tout ce qui ne se formalise pas ou ne se transmet pas de façon immédiate, rend malaisé le traitement et la transmission systématique et logique de celles-ci.

Le développement de la connaissance dans les organisations implique la transmission entre les individus de celle-ci ainsi qu'une dynamique de transformation entre différents états. Cette dynamique de transformation signifie que la connaissance peut passer dans différents états suivant des mécanismes que nous allons expliquer par la suite. Selon Nonaka et Takeuchi, il existe quatre manières différentes de réaliser la transformation entre les connaissances tacites et explicites donnant lieu à quatre états

différents. Nous retrouverons ces quatre états dans le SCHEMA 1.1. Les explications des quatre transformations se basent sur le livre intitulé *'La connaissance créatrice'*. [NONAKA97]

La socialisation représente la transformation du tacite vers le tacite. Elle consiste en l'interaction d'individus au sein d'un groupe. Celle-ci découle de l'apprentissage implicite qui se fait par l'observation consciente ou inconsciente, l'imitation, le partage d'expérience, etc. Ainsi, les apprentis travaillent avec leurs maîtres en apprenant la connaissance d'un métier non par le langage mais par l'observation, l'imitation et la pratique.

L'extériorisation, également nommée **la formalisation**, représente la transformation du tacite vers l'explicite. Elle repose sur l'explicitation, par le discours ou l'écrit, des pratiques et des croyances. La difficulté réside dans l'adoption d'un langage et de concepts partagés. Ainsi, il est possible de transformer la connaissance tacite en connaissance explicite grâce à des techniques telles que la métaphore, l'analogie, l'induction ou la déduction.

L'intériorisation représente la transformation de l'explicite vers le tacite. Elle consiste en l'enracinement de la connaissance explicite dans des séquences pouvant atteindre le stade du réflexe, de l'automatisme, et devant normalement s'accompagner d'une efficacité. L'utilisation du coaching et des systèmes d'aide à l'apprentissage permet en effet de faciliter l'apprentissage dans l'action de nouvelles pratiques.

La combinaison représente la transformation de l'explicite vers l'explicite. Elle consiste en la communication des connaissances explicites qui sont combinées, rapprochées pour produire, par induction et/ou déduction, des nouvelles connaissances. Nous retrouvons ce principe dans les forums d'experts.

	Connaissance tacite	Connaissance explicite
Connaissance tacite	**Socialisation**	**Extériorisation**
Connaissance explicite	**Intériorisation**	**Combinaison**

SCHEMA 1.1 : Modes de conversion des connaissances.[NONAKA97, p83]

2. Le concept de gestion des connaissances

Après avoir identifié la signification des différents concepts qui sont à la base de la gestion des connaissances, nous allons maintenant voir ce que l'on entend par la gestion des connaissances. Il faut remarquer que ce concept de gestion des connaissances se fonde sur deux termes assez abstraits : d'une part, la gestion, et, d'autre part, la connaissance. Il est donc difficile d'en donner une définition spécifique. C'est pourquoi nous commencerons par citer quelques définitions provenant de différents auteurs. Nous préciserons par la suite ce que nous entendons par gestion des connaissances et ce que nous devons retrouver dans celle-ci.

Selon Bouteillier, « la gestion des connaissances est une nouvelle science visant à réorganiser l'entreprise autour de sa richesse immatérielle ». [JACOB00, p23]

Hamilton considère, quant à lui, que « la gestion des connaissances est un processus de création, d'acquisition, de transfert et d'utilisation des connaissances dans le but d'améliorer le rendement de l'organisation ». [JACOB00, p23]

Rossett défend, quant à lui, l'idée selon laquelle le « Knowledge management involves recognising, documenting and distributing explicit and tacit knowledge in order to improve organizational performance ». [JACOB00, p23]

Selon nous, la gestion des connaissances poursuit trois buts. Elle vise premièrement à collecter les connaissances disponibles dans l'organisation que celles-ci soient des connaissances explicites ou tacites. Ensuite, elle a pour tâche d'assurer la production des nouvelles connaissances ; connaissances qui sont alors considérées comme étant collectives. Elle permet enfin l'identification, la codification et la diffusion des connaissances.

La gestion des connaissances est donc avant tout une démarche organisationnelle. Elle permet en théorie de faire progresser le savoir-faire de l'entreprise, de favoriser la communication et d'entreprendre avec succès des changements organisationnels de grande envergure.

On se retrouve ainsi dans une situation où la plupart des organisations cherchent à innover pour garantir leur compétitivité. Afin de permettre une innovation à faible coût, on doit s'assurer de la maîtrise des connaissances au sein de l'organisation et du partage de celles-ci entre les différents acteurs de l'organisation. Comme le dit Jacob, « la capacité innovante d'une organisation serait donc ancrée dans sa capacité à transformer ses actifs de connaissances plus ou moins organisées et individuelles en 'intelligence stratégique collective'». [JACOB00, p12]

II. Raisons de l'engouement pour la gestion des connaissances

Pourquoi, dans les années 90, remarque-t-on une expansion du concept de gestion des connaissances au sein des organisations ? Quelles en sont les causes ?

1. La mondialisation

Il faut être conscient que, dans les années 90, les organisations se retrouvent dans un environnement caractérisé par la mondialisation de l'économie. Celle-ci a pour effet de rendre la concurrence de plus en plus apparente et prégnante. Les différentes organisations sont donc confrontées à une situation dans laquelle il devient vital de maîtriser la complexité des connaissances et des technologies pour être plus compétitif. Le monde des

organisations est donc tout doucement en train de passer d'une logique d'affaire basée sur les ressources à une logique d'affaire fondée sur les connaissances. Comme le disent Nonaka et Takeuchi en précisant la pensée de Drucker, « (...) dans la nouvelle économie, la connaissance n'est pas seulement une nouvelle ressource qui s'ajoute aux facteurs de production traditionnels comme le travail, le capital, la terre, mais la seule ressource qui ait une signification réelle de nos jours. Le fait que la connaissance soit devenue la ressource plutôt qu'une ressource est, selon lui, le trait distinctif de la nouvelle société. ». [NONAKA97, p24] De plus, l'aptitude à disposer de la bonne connaissance au bon endroit au bon moment est elle aussi un facteur de réussite des organisations.

2. Le turnover des employés

Un autre facteur explicatif de l'engouement pour la gestion des connaissances souvent cité est le turnover des employés de l'organisation. Cette notion englobe plusieurs cas : premièrement, le cas dans lequel un employé quitte une organisation pour aller travailler dans une autre organisation. Cette situation est relativement fréquente dans les entreprises qui sont fortement orientées vers le monde des hautes technologies où un employé fait sa carrière dans plusieurs entreprises. Deuxièmement, le cas dans lequel un employé quitte son emploi pour prendre sa retraite. Cette situation se retrouve de plus en plus souvent car la tendance actuelle est de prendre sa retraite de plus en plus jeune. Troisièmement, le cas dans lequel un employé change de projet au sein de la même organisation.

Le problème n'est pas le fait que l'employé quitte physiquement l'organisation, mais plutôt le fait qu'il parte avec toutes ses connaissances. Or ses connaissances peuvent être très utiles pour le reste de l'organisation ou plus précisément pour le reste de l'équipe de projet.

Un moyen de pallier aux inconvénients du turnover des employés est de stocker et de rendre disponibles, au jour le jour, les connaissances individuelles et locales dans une mémoire organisationnelle. Mais cette solution ne risque-t-elle pas justement d'inciter les directeurs, une fois les connaissances des employés stockées, à renvoyer ceux-ci ?

3. La perte de temps

Une autre raison pour laquelle les organisations recourent à la gestion des connaissances repose sur une étude d'EDS réalisée auprès des mille plus grandes entreprises américaines. Celle-ci nous signale qu'en 1996, les employés passaient en moyenne 60 % de leur temps à rechercher et valider de l'information. Pour limiter cette perte de temps, il faudrait que l'employé puisse identifier les informations et les interpréter en connaissances, pour ensuite les codifier et les diffuser à toute l'organisation.

Voyons maintenant ce que nous disent les sondages réalisés dans les années 90 en ce qui concerne l'importance que les organisations accordent à la gestion des connaissances. (TABLEAU 1.1) Identifions ensuite la pénétration de la gestion des connaissances au sein de nos organisations. (TABLEAU 1.2)

Importance de la gestion des connaissances selon divers sondages
• 79 % des PDG interrogés estiment que la gestion du savoir est vitale pour le succès de leur entreprise. *(American Management Association, 1999)*
• Pour la très grande majorité des dirigeants d'entreprises européennes interrogés, la gestion du savoir est un facteur critique pour augmenter les profits, accroître ses avantages compétitifs et réussir dans le futur. *(Granfield School of Management, 1998)*
• « La gestion du savoir est vitale pour le succès futur de notre entreprise ». *(American Management Association, 1998)*
• Une récente étude de benchmarking en Europe indique que plus de 50 % des 'best practice organizations' incluent explicitement la gestion du savoir dans leurs objectifs stratégiques. *(American Productivity and Quality Center, cité par Competitive Intelligence Magazine, 1999)*
• Pour 94 % des gestionnaires en système d'information sondés, la gestion du savoir est un enjeu stratégique en premier ordre pour l'entreprise. *(Information Week Research, 1999)*
• 90 % des gestionnaires de ressources humaines croient que le capital intellectuel est un enjeu de plus en plus important. *(National HRD, 1997)*

TABLEAU 1.1 : Importance de la gestion des connaissances. [JACOB00, p14]

Taux de pénétration de la gestion des connaissances selon divers sondages
• 60 % des organisations de Boston situent leurs efforts en gestion des connaissances dans la moyenne ou en dessous. 28 % ont déjà mis en place une initiative formelle et 93 % ont l'intention d'en développer une au cours de l'an 2000. *(Delphi Consulting Group 1997)*
• 43 % des répondants déclarent mettre de l'avant quelques initiatives formelles en matière de gestion des connaissances. Cependant encore un tiers des entreprises emmagasinent les informations sous des formats non technologiques comme le cerveau des individus ou le papier. *(KPMG, 1998)*
• En Angleterre, bien que 66 % des entreprises sondées considèrent que l'information est un élément critique dans les affaires, 39 % seulement ont donné formellement une suite. *(Interactive Information Service, 1998)*
• 36 % des répondants disposent de systèmes formels pour partager les connaissances, mais que ce pourcentage est plus faible lorsqu'il est question des processus formels d'identification et de codification des connaissances. 62 % des organisations interrogées disent qu'elles investiront plus d'efforts dans la lignée de la gestion des connaissances plus tard. *(Davis and Riggs, 1999)*

TABLEAU 1.2 : Pénétration de la gestion des connaissances. [JACOB00, p18]

Ces quelques données chiffrées montrent que même si la gestion des connaissances semble une préoccupation principale de la majorité des organisations, concrètement presque rien n'est vraiment mis en œuvre pour répondre à ce besoin. On peut donc dire que la gestion des connaissances est de nos jours à un stade embryonnaire. Mais pourquoi est-elle si peu développée ? A notre avis, tout simplement parce que, même si les connaissances présentent intrinsèquement une grande richesse, celles-ci restent assez complexes et multiformes. Il est donc difficile de les gérer de manière optimale. Nous verrons dans les chapitres suivants qu'il existe des méthodes pour gérer la connaissance ainsi que des outils comme celui que nous détaillerons dans la seconde partie du présent travail.

III. Buts de la gestion des connaissances

Un des gros problèmes qui est souvent repris par les directeurs d'entreprises se retrouve sous cette forme : « Si nous savions tout ce que l'on sait ! ». Ce problème provient d'un manque de communication au sein de l'organisation. On remarque en effet que l'organisation est généralement composée de multiples connaissances individuelles et non d'une connaissance collective (connaissance organisationnelle).

Il est vrai que certaines organisations, comme les banques, sont dotées d'une mémoire collective dans laquelle elles reprennent toutes les procédures régies par l'organisation. Par contre, d'autres organisations ne croient pas au bien fondé des mémoires collectives car elles considèrent qu'il peut être difficile de passer d'une culture organisationnelle individuelle à une culture organisationnelle collective. Nous reviendrons sur ce point plus tard dans ce présent chapitre. Mais quels sont les buts de la gestion des connaissances ?

1. Les différents buts

Selon Ermine, les objectifs de la gestion des connaissances peuvent être décrits sur base de trois mots clefs : **capitaliser**, c'est-à-dire selon lui « Savoir d'où l'on vient, savoir où l'on est, pour mieux savoir où l'on va » ; **partager**, c'est-à-dire selon lui « Passer

de l'intelligence individuelle à l'intelligence collective » et **créer**, c'est-à-dire selon lui « Créer, innover pour survivre ». [ERMINE96]

Selon Jacob, en précisant l'idée de Davenport, nous retrouvons dans la gestion des connaissances trois noyaux qui sont **la génération, la codification et la diffusion**. Ces trois noyaux doivent obligatoirement être les maîtres mots de la gestion des connaissances. **La génération** représente le processus permettant d'identifier toutes les connaissances, le capital intellectuel. Ce processus offre également la possibilité de créer de nouvelles connaissances collectives. **La codification**, quant à elle, permet de codifier les connaissances dans un langage commun et ainsi de les rendre accessibles à tous les individus de l'entreprise. **La diffusion** permet de transférer les connaissances codifiées pour les rendre accessibles à tous les membres du personnel. [JACOB00]

D'après Bès, la gestion des connaissances suit deux orientations. La première repose sur le fait de repêcher les technologies, afin de remobiliser les connaissances et de retrouver des éléments du contexte de travail. Il s'agit dans cette orientation de comprendre les choix et les erreurs antérieurs. La seconde réduit les pertes d'informations lors de la conduite des projets et prévoit, au départ du projet, de transmettre certaines connaissances en effectuant un effort d'extraction de ces connaissances et d'explication du contexte. [BES98]

Dans notre conception, la gestion des connaissances s'apparente à un procédé qui se rapproche plus de l'idée de Davenport que de celle de Bès. Pour nous, nous devons en effet obligatoirement retrouver dans la gestion des connaissances la codification des connaissances suivant un procédé toujours identique ainsi qu'un moyen de diffusion de celles-ci pour les rendre accessibles à tout le monde. A l'opposé de Davenport, nous ne considérons pas la génération comme un noyau principal mais plutôt comme un noyau qui se mettra en place une fois que l'outil de gestion des connaissances sera bien installé et bien maîtrisé. Par contre, il est vrai que nous devons percevoir cet outil non comme un moyen de stockage (donc d'archivage) des connaissances mais plutôt comme un flux de connaissances. Il doit dès lors être possible de faire des liens entre les différentes connaissances insérées dans le système.

En résumé, nous pouvons donc dire que la gestion des connaissances aura comme objectifs les différents buts qui vont suivre.

Le but premier de la gestion des connaissances se situe évidemment sur un plan financier. Les organisations, comme nous l'avons vu précédemment, recherchent une meilleure position compétitive qui passe par une plus grande innovation. Pour atteindre ces objectifs, il faut également diminuer les coûts de production. Une manière d'y parvenir est la gestion des connaissances.

La gestion des connaissances devra donc créer des opportunités de collaboration entre employés et cette collaboration aura pour effet de générer de nouvelles connaissances qui seront alors des connaissances collectives.

Il est également nécessaire que la gestion des connaissances permette aux employés de trouver facilement une réponse à un problème. Surtout si ce problème a déjà été résolu par d'autres employés dans un département différent de l'entreprise ou au sein du même département.

La gestion des connaissances devra également favoriser les connaissances collectives par rapport aux connaissances individuelles. Elle doit donc privilégier la mémoire organisationnelle de l'entreprise.

La gestion des connaissances veillera à permettre une codification aisée des différentes connaissances détenues par un individu de l'organisation. Une fois cette étape réalisée, l'organisation permettra une bonne diffusion de celles-ci en son sein.

2. Difficultés pour atteindre ces buts

La question est maintenant de savoir comment réussir à obtenir ces deux noyaux que sont la codification et la diffusion. Il nous semble raisonnable de dire que l'informatique est un des moyens à envisager. Nous pourrions évidemment imaginer une solution basée sur le support papier dans laquelle chaque employé formaliserait ses

connaissances sur papier et procurerait une copie de celles-ci à ses collègues. Mais cette solution ne présente pas une sécurité et une rapidité exemplaire.

Cependant, l'informatique n'est pas la solution aux problèmes. L'informatique n'est qu'un moyen, un outil pour permettre d'atteindre les objectifs de la gestion des connaissances. A la base de l'informatique, nous devons retrouver une méthode permettant de collecter les connaissances. Nous aborderons ces méthodes dans le chapitre suivant. La limite de l'informatique se retrouve dans son incapacité de formaliser les connaissances tacites alors que celles-ci sont celles qui vont justement permettre l'innovation.

Maintenant, une fois une méthode de collecte des connaissances et une stratégie de gestion des connaissances établies, l'informatique permettra sans doute de codifier les connaissances ainsi que d'offrir la possibilité d'accéder aux connaissances plus rapidement en éliminant une partie du travail de recherche nécessaire pour retrouver la connaissance désirée. Elle permettra aux employés de mettre à jour les connaissances et de les améliorer aisément ainsi que d'établir des liens entre elles si nécessaire.

Mais, il est important de noter qu'il persiste un problème : la surinformation ou ce que l'on pourrait appeler 'l'infobésité'. En effet, il n'est jamais bon pour un individu de se retrouver face à une surcharge de connaissances. Cette surcharge se remarque clairement quand on travaille avec un support papier (l'on se perd sous un amas de feuilles). Cette surcharge peut aussi se rencontrer avec un support informatique si la présentation des connaissances n'est pas standardisée et si ces connaissances ne sont pas correctement mises en relation. Il faut donc, lors de la création d'un outil de ce genre, veiller à bien présenter les connaissances et à avoir un flux entre celles-ci.

IV. Obstacles et avantages de la gestion des connaissances

1. Les obstacles

Le premier obstacle que les organisations rencontreront est humain et très difficile à résoudre. Il s'agit tout simplement d'une résistance des employés à divulguer leurs connaissances à d'autres puisque celles-ci leur permettent justement d'affirmer leur

position dans l'organisation. Pour bien faire, il faudrait que l'on se retrouve dans 'une organisation apprenante' et que chaque employé soit conscient de ce régime et y adhère. Ainsi, selon Tarondeau « Certaines organisations apprennent mieux et plus que d'autres. Certaines exploitent les savoirs qu'elles génèrent de façon plus efficace que d'autres. Les organisations apprenantes s'efforcent de réussir sur ces deux dimensions. » [TARONDEAU98, p88]

La codification des connaissances tacites constitue un autre obstacle. En effet, il est assez aisé de formaliser des connaissances explicites. Par contre, les connaissances tacites sont immatérielles et ancrées à l'intérieur de l'individu sans qu'il s'en rende vraiment compte. Il est donc recommandé de favoriser le principe d'extériorisation vu précédemment.

Un autre obstacle se situe plus sur le plan financier. En effet, un procédé de gestion des connaissances peut coûter cher et il est difficile d'évaluer exactement la somme que ce procédé va rapporter à l'organisation. Et l'on sait bien que les chefs d'entreprises, même si le projet est cohérent, n'investissent pas facilement dans un projet dont ils ne savent pas estimer exactement le bénéfice futur.

Il faut également que les divers employés puissent à la fois accéder facilement aux connaissances et les encoder simplement. Pour cela il ne faut pas que le système soit trop éloigné de leurs habitudes. Ceux-ci doivent se retrouver devant un outil avec une interface 'user-friendly' que ce soit pour l'encodage ou pour la recherche des connaissances.

2. Les avantages

Il est évident que l'avantage premier est le fait qu'en assurant la gestion des connaissances l'organisation peut améliorer sa position concurrentielle et ses innovations. Un avantage à plus long terme serait celui de permettre à une organisation ayant une bonne gestion de ses connaissances d'obtenir une certification ISO9000 pour renforcer sa position compétitive.

La gestion des connaissances et la production de nouvelles connaissances collectives permettent l'amélioration de la productivité de l'entreprise et la performance des processus. En effet, elles améliorent les démarches de résolution de problèmes par les employés de l'organisation. Chaque employé est plus productif car il a en sa possession un éventail de connaissances beaucoup plus large.

La gestion des connaissances permet de maîtriser et d'améliorer la qualité des produits ou des services rendus. En effet, en maintenant ses connaissances à jour, l'organisation peut utiliser les connaissances les plus adéquates pour créer le produit ou pour rendre le service correspondant à la demande du client.

Dernier avantage qui n'est pas des moindres, la gestion des connaissances permet de passer d'une organisation à connaissances individuelles, c'est-à-dire d'une organisation qui ne vit pas en elle-même mais qui vit grâce aux connaissances détenues par ses propres employés, à une organisation dans laquelle prévaut la mémoire d'entreprise. On parlera alors de connaissances organisationnelles et l'organisation ne sera plus vue comme un ensemble d'employés différents les uns des autres mais comme une entité en elle-même.

Conclusion de chapitre

En conclusion, nous avons précisé dans ce premier chapitre ce que l'on entendait par la gestion des connaissances. Nous avons constaté qu'il était difficile de donner une définition précise d'un concept qui englobe deux concepts assez abstraits. Dès lors, la gestion des connaissances doit à notre sens tenir compte de plusieurs éléments. Premièrement, il est nécessaire qu'elle collecte les connaissances disponibles dans l'organisation, que ce soit des connaissances explicites ou tacites. Deuxièmement, il convient d'assurer la production de nouvelles connaissances qui sont alors considérées comme étant collectives. Troisièmement, il faut permettre l'identification, la codification et la diffusion des connaissances. La gestion des connaissances se résume donc en trois mots empruntés à Ermine : CAPITALISER, PARTAGER et CREER.

Les raisons de cet engouement pour la gestion des connaissances sont multiples. La raison principale est la recherche d'une meilleure compétitivité dans un environnement

où les organisations sont confrontées à une concurrence de plus en plus accrue. On essaye donc dans une organisation de faire en sorte qu'un employé ait à sa disposition la bonne connaissance au bon endroit et au bon moment. Il faut aussi combattre la perte des connaissances et donc favoriser une mémoire organisationnelle.

Les entreprises recherchent de nos jours des méthodes ou des outils qui permettent de collecter les connaissances de chacun, de les diffuser au sein de l'organisation toute entière. Il s'agit donc de réunir et de partager des connaissances collectives et non individuelles. Pour cela, il nous semble évident que l'informatique est un élément moteur, que ce soit pour la collecte ou pour la diffusion, car elle remplace la profusion de documents sur support papier au sein de l'organisation.

Pour atteindre ses objectifs, la gestion des connaissances risque de rencontrer quelques obstacles dont le principal est d'ordre humain. Cependant, la gestion des connaissances peut apporter beaucoup de bien à une organisation. L'avantage principal à pointer réside dans la possibilité pour la gestion des connaissances d'apporter un avantage concurrentiel à l'organisation.

Nous allons maintenant voir les différentes méthodes qui existent pour gérer correctement les connaissances des employés d'une organisation. Il est à noter que dans le chapitre suivant nous parlerons de méthodes et non d'outils même si certaines méthodes sont dotées de logiciels. Nous aborderons rapidement ces outils liés à ces méthodes de gestion des connaissances et nous parlerons plus spécifiquement, dans la deuxième partie de ce présent travail, de l'outil que nous avons en partie développé.

Deuxième chapitre

Méthodologie concernant la gestion des connaissances

I. Méthode empirique
    1. Schéma général
    2. Explication des différentes phases
        2.1. Recueil de l'expertise
        2.2. Interprétation et modélisation de l'expertise
II. Méthodes construites
    1. MACAO
        1.1. En théorie
            1.1.1. La détermination des matériaux de base
            1.1.2. La construction du schéma du modèle conceptuel
            1.1.3. La définition du modèle conceptuel complet
            1.1.4. La validation du modèle conceptuel complet
        1.2. En pratique
            1.2.1. Quels sont les objectifs d'une résolution de problèmes ?
            1.2.2. Quels sont les moyens pour atteindre ces objectifs ?
            1.2.3. Quelles sont les connaissances du domaine manipulées ?
    2. KADS
        2.1. En théorie
            2.1.1. Le modèle d'expertise
            2.1.2. Le modèle organisation
            2.1.3. Le modèle tâche
            2.1.4. Le modèle agent
            2.1.5. Le modèle communication
            2.1.6. Le modèle de conception
        2.2. En pratique
    3. REX
        3.1. En théorie
        3.2. En pratique
    4. MKSM
        4.1. En théorie
            4.1.1. Le modèle du système de référence
            4.1.2. Le modèle du domaine
            4.1.3. Le modèle d'activité
            4.1.4. Le modèle des concepts
            4.1.5. Le modèle des tâches
        4.2. En pratique
III. Fiches d'identité

Introduction de chapitre

Nous allons dans ce deuxième chapitre aborder plusieurs méthodes consacrées à la gestion des connaissances. Les méthodes décrites dans ce présent chapitre sont toutes, avant tout, des méthodes et non des outils de gestion des connaissances même si certaines ont été complétées par un outil logiciel. Les différentes méthodes qui suivront, du moins celles que nous dénommerons 'les méthodes construites', ont toutes fait leurs preuves dans un domaine particulier et sont encore de nos jours utilisées dans différentes organisations.

Les 'méthodes construites' ont vu le jour en s'inspirant de la méthode empirique, mais elles sont plus rigoureuses que cette dernière, et ce, que ce soit dans l'acquisition ou dans la diffusion des connaissances. Nous ne présenterons pas, dans ce présent travail, toutes les 'méthodes construites' existantes. Nous nous limiterons aux méthodes les plus fréquemment présentées par les auteurs de la gestion des connaissances. Nous aborderons donc les méthodes MACAO (Méthode d'Acquisition des Connaissances Assistée par Ordinateur), KADS (Knowledge Analysis and Design System), REX (Retour d'EXpérience) et MKSM (Method for Knowledge System Management) tout en reconnaissant qu'il en existe bien d'autres comme les méthodes SAGACE, MEREX (Mise En Règles de l'EXpérience), etc. Il faut noter dès à présent que ces méthodes diffèrent par leur domaine d'application spécifique.

Concrètement, nous présenterons dans un premier temps la méthode empirique de gestion des connaissances. Deuxièmement, nous parlerons des méthodes que nous avons nommées 'méthodes construites'. Au sein de celles-ci, nous aborderons les méthodes MACAO, KADS, REX et MKSM ; l'ordre logique de présentation de ces dernières étant dicté par la chronologie de leur apparition. Troisièmement, nous présenterons les fiches d'identité des différentes méthodes. Ces fiches reprendront les points importants des méthodes décrites.

I. Méthode Empirique

Comment une organisation peut-elle œuvrer pour récolter les connaissances de ses employés ? La première méthode que nous allons envisager est celle que Bès a appelé 'la méthode du biseau'.[BES98] Cette méthode consiste à faire travailler ensemble deux employés pendant une période déterminée, l'un quittant son emploi, l'autre le remplaçant. Le but est bien entendu que l'employé sortant partage son savoir et son savoir-faire qu'il a acquis au sein de l'organisation pour que l'employé entrant puisse travailler dans les meilleures conditions. Cette méthode, qui est assez générique, sera appelée dans ce travail la méthode empirique. Elle consiste donc en l'échange de connaissances entre d'une part un expert et d'autre part un novice capable de réaliser un recueil de connaissances. La méthode empirique constitue le fondement des méthodes que nous avons appelées 'les méthodes construites'.

Plus généralement, la méthode empirique se base sur la rencontre de deux individus, un expert et un novice (ou un interviewer) qui est là pour récolter les connaissances de ce premier afin de créer un système à base de connaissances. Cette méthode empirique comprend plusieurs phases dont la première est le recueil des informations et la dernière la programmation des connaissances collectées.

1. Schéma général

Le recours à la méthode empirique nécessite la collaboration de l'expert et du novice suivant quatre phases.

a) Recueil des informations à partir des connaissances documentées et du savoir-faire de l'expert.
b) Création d'un dossier d'expertise à partir du recueil d'informations.
c) Interprétation et modélisation de l'expertise (le recours à un informaticien peut être très utile à ce niveau).
d) Application d'un système à base de connaissances.

2. Explication des différentes phases

Méthodologiquement, nous avons décidé de regrouper d'une part les deux premières phases et d'autre part les deux dernières. Ce regroupement s'explique par le fait que les deux premières phases se réalisent une et une seule fois alors que les deux dernières forment un cycle et peuvent donc se répéter.

2.1. Recueil de l'expertise

Le novice récolte les connaissances documentées et le savoir-faire de l'expert et les réunit dans un recueil. Pour ce faire, le novice utilise des techniques peu formalisées comme des entretiens peu structurés et l'observation de l'expert dans son milieu de travail. Cependant, la présence du novice peut troubler l'expert et dès lors, ce dernier peut ne pas agir comme à son habitude. En effet, il est assez troublant pour un individu de se sentir épié que ce soit par un autre individu ou une caméra. L'employé sera tenté, durant l'observation, de travailler suivant les directives de l'organisation et non plus en suivant ses habitudes, ses propres règles qui peuvent parfois être plus optimales que celles proposées par l'organisation.

2.2. Interprétation et modélisation de l'expertise

Après avoir élaboré le recueil d'informations et le dossier d'expertise, le novice doit traduire le dossier en règles. Lorsqu'il a réalisé la traduction, il propose ses règles à l'expert qui aura pour but de les valider si celles-ci représentent correctement ses connaissances ou au contraire de les refuser si ses connaissances ont été mal traduites. Ce processus forcera ainsi le novice à proposer de nouvelles règles tant que l'expert ne les aura pas validées. Nous nous trouvons donc bien en présence d'un cycle.

Les limites de cette méthode sont nombreuses. En effet, elle peut entraîner une déformation des connaissances de l'expert et présenter une certaine incomplétude. Effectivement, rien ne garantit au novice que l'expert lui a parlé de toutes ses

connaissances et a usé de tous ses savoir-faire. De plus, cette méthode s'avère relativement longue et coûteuse.

Pour combler ces lacunes, nous allons maintenant étudier quelques méthodes que nous appellerons 'méthodes construites'. Certaines sont encore utilisées de nos jours.

II. Méthodes construites

Dans cette partie, nous verrons des méthodes qui ont fait leurs preuves en matière de gestion des connaissances. Celles-ci seront présentées selon un ordre logique déterminé par leur année de création. Nous commencerons par MACAO qui permet d'extraire les connaissances et de les organiser avec l'aide de l'expert. Nous aborderons ensuite la méthode KADS qui aide à la modélisation des connaissances d'un expert ou d'un groupe d'experts dans le but de réaliser un système d'aide à la décision basé sur la connaissance. Ensuite, viendra REX, qui est une méthode permettant de capitaliser les connaissances et de favoriser le retour d'expérience. Nous parlerons finalement de MKSM, une méthode permettant de maîtriser la complexité dans les projets de gestion des connaissances avant d'aboutir à un projet opérationnel.

1. MACAO

La méthode MACAO (Méthode d'Acquisition des Connaissances Assistée par Ordinateur) est développée depuis 1988. Elle est le résultat de la thèse de doctorat de Aussenac-Gilles qui travaille à l'IRIT (Institut de Recherche en Informatique de Toulouse). Cette thèse défendue en octobre 1989 s'intitule *'Conception d'une méthodologie et d'un outil d'acquisition des connaissances expertes'*. Cette méthode a pour objectif de fournir au novice un support méthodologique et logiciel pour l'acquisition des connaissances. Dès lors, MACAO est à la fois une méthode et un outil logiciel.

Quel en est le fonctionnement ? Selon Aussenac-Gilles, « MACAO permet de construire un modèle décrivant l'expertise nécessaire à réaliser la tâche assignée au futur

système à base de connaissances. Ce modèle est défini comme un modèle conceptuel servant de support au dialogue entre cogniticien et expert. Il est construit et visualisé à l'aide du logiciel associé à la méthode et composé de connaissances structurées de manière plus ou moins formelle. Il doit ensuite être traduit pour construire une base de connaissances ». [AUSSENAC1] En bref, MACAO fournit des techniques et des recommandations pour recueillir et analyser les connaissances expertes pour en dégager un modèle peu formel.

Cette méthode sera par la suite remodelée par Matta en la méthode MACAO-II. Cette dernière permet une évolution de la méthode d'acquisition et de modélisation des connaissances en favorisant entre autre la représentation des connaissances et des méthodes grâce au langage MONA.

1.1. En théorie

Comme pour la méthode empirique, MACAO s'exécute en quatre phases : la détermination des matériaux de base, la construction du schéma du modèle conceptuel, la définition du modèle conceptuel complet et la validation du modèle conceptuel complet.

1.1.1. La détermination des matériaux de base

Il faut avant tout que le novice et l'expert se mettent d'accord sur la terminologie. Il convient également que l'expert soit en confiance afin d'éviter tout malentendu entre ces deux acteurs. Pour déterminer les matériaux de base, le novice commence par identifier l'environnement de l'expertise, par expliciter la terminologie, et par essayer de comprendre la démarche générale de résolution de problèmes de l'expert.

Les techniques employées pour cette phase sont des entretiens informels, une observation directe sur le lieu de travail de l'expert, des entretiens centrés sur un type de problème et enfin la mise en œuvre par le logiciel de grilles répertoires pour séparer les différents types de problème.

L'intérêt de cette phase est de permettre au novice de réunir un ensemble très riche de données de base, qui constituent une sorte de photographie très complète de l'expertise. Le novice dispose donc de détails pour bien comprendre la résolution générale de problèmes. Malheureusement, ces données sont difficiles à structurer car de bas niveau.

1.1.2. La construction du schéma du modèle conceptuel

Le but de la seconde phase, concernant la construction du schéma du modèle conceptuel, est d'élaborer des protocoles de résolution de problèmes ainsi que des explications sur ces mêmes résolutions. Concrètement, il s'agit d'obtenir une première formalisation de la structure du modèle conceptuel.

Qu'est ce qu'un modèle conceptuel ? Un modèle conceptuel est un support qui sert à la fois à traduire le plus précisément possible les connaissances de l'expert, tout en facilitant leur expression et leur structuration. Malheureusement, ces modèles ne sont pas directement opérationnels. De plus, ils sont spécifiques à chaque expertise considérée et dès lors non réutilisables.

Nous retrouvons dans le schéma du modèle conceptuel deux parties : d'une part, la modélisation du domaine et d'autre part, la modélisation du raisonnement.

La modélisation du domaine

Les connaissances du domaine sont représentées par des concepts génériques et des relations entre ces concepts. L'ensemble des concepts et des relations forme un réseau qui peut être représenté sous forme de graphes.

La modélisation du raisonnement

Le raisonnement est représenté par un enchaînement de schémas. La structure de ces schémas a été définie pour représenter un pas de raisonnement associé à la réalisation d'un but. Le raisonnement général s'exprime par un enchaînement de schémas qui

constitue le modèle de résolution du problème ou de l'expertise. Ce raisonnement général est représenté sous forme d'arbre.

1.1.3. La définition du modèle conceptuel complet

Le but de cette phase est à nouveau de récolter des données. Cependant, dans cette phase, la récolte se fait dans un cadre bien précis qui est fourni par le modèle conceptuel. Il faut donc selon Aussenac-Gilles « acquérir, conceptualiser et représenter toutes les connaissances du domaine prévues par le modèle : les instances de concept, leurs valeurs, les règles et heuristiques correspondant à des pas de raisonnement ou à des méthodes du modèle, etc. Le modèle est considéré comme complet lorsque toutes les connaissances devant jouer les rôles prévus par le modèle ont été identifiées. ». [AUSSENAC2]

1.1.4. La validation du modèle conceptuel complet

Cette dernière phase permet de valider la fonctionnalité du modèle conceptuel créé. Elle consiste tout simplement à rendre opérationnel le modèle en l'introduisant dans un langage spécifique. Ceci pour vérifier les capacités de résolution de problèmes du modèle conceptuel et la qualité des solutions produites.

1.2. En pratique

La représentation des connaissances de résolution de problèmes dans un modèle conceptuel doit répondre à deux types de besoins : nous devons pouvoir interpréter ce modèle en lui attribuant un sens et ce modèle doit faciliter la programmation d'un système basé sur les connaissances. C'est pourquoi est apparu le langage MONA au sein de la méthode MACAO-II. Ce langage permet d'évoluer progressivement d'une représentation informelle vers une représentation formelle des connaissances recueillies. Nous allons montrer l'apport de MONA dans différentes sous tâches de la modélisation telles que le recueil, la structuration et le rendu opérationnel des connaissances.

Pour ce faire, nous allons essayer de formaliser le comportement d'un expert en répondant aux trois questions suivantes.

1.2.1. Quels sont les objectifs d'une résolution de problèmes ?

MONA définit ce qui est appelé 'la structure tâche'. Cette structure permet d'exprimer le but à atteindre, les contraintes que doivent vérifier les connaissances, les critères de satisfaction de la tâche, le contexte dans lequel celle-ci évolue ainsi que les contraintes sur les éléments de ce contexte et une déclaration des méthodes susceptibles d'accomplir la tâche.

1.2.2. Quels sont les moyens pour atteindre ces objectifs ?

MONA définit ce qui est appelé 'la structure de méthode'. Cette structure permet de décrire les paramètres d'entrée d'une méthode, les résultats, les contraintes que les connaissances en entrée doivent vérifier et le traitement que la méthode doit effectuer pour fournir les résultats. Ce traitement peut être une décomposition en un certain nombre de tâches à réaliser aussi bien qu'une simple procédure.

1.2.3. Quelles sont les connaissances du domaine manipulées ?

Les connaissances du domaine sont des concepts. MONA définit un concept dans une structure propre dans laquelle les propriétés de ce concept sont représentées à l'aide d'attributs. Les relations entre ce concept et les autres concepts ou attributs sont aussi associées à ce concept ainsi que l'ensemble des rôles qu'il joue dans la résolution de problèmes.

En conclusion, pour une expertise donnée, les structures MONA sont organisées en un modèle du domaine formé des concepts, des liens et des propositions ainsi qu'un modèle du raisonnement décrit à l'aide des tâches et des méthodes. Ces structures forment une représentation formelle des connaissances recueillies. Il faut maintenant coder cette représentation. Ce codage se déroule en deux phases : la première consiste en une

traduction automatique de cette représentation dans des structures LISA (LISA est une sur-couche du langage LE_LISP), la seconde consiste en un codage manuel qui vient enrichir et compléter les structures LISA générées.

L'intérêt primordial de cette méthode prend sa source dans le modèle conceptuel. En effet, celui-ci permet de modéliser à la fois les connaissances du domaine et le raisonnement de l'expert. Ce modèle conceptuel est facilement formalisable. Ceci permet de suivre une certaine structure avant la formalisation des connaissances et d'éviter ainsi l'anarchie d'une formalisation trop directe.

2. KADS

La méthode KADS (Knowledge Analysis and Design System) a été conçue en 1989 par Brooking, Breuker, Wielinga et Rogers dans le cadre d'un programme européen intitulé Esprit I. Plus tard, à partir des années 90, pour le programme Esprit II est apparue la méthode KADS-II dans le but d'en faire un standard commercial européen. Selon Barthes, cette méthode a pour but d'essayer « (...) de modéliser les stratégies de raisonnement d'un spécialiste de façon abstraite et de développer une bibliothèque d'actions génériques faisant intervenir une modélisation des connaissances stratégiques et des connaissances du domaine. ». [BARTHES98]

La méthode KADS s'organise en deux grandes phases. La première est appelée la phase d'analyse. Elle monopolise l'expert et regroupe tout ce qui a trait au recueil de connaissances. Une fois cette phase terminée, on passera à la seconde phase et on ne reviendra plus à la première. La seconde est appelée la phase de conception, elle est réalisée par le novice qui doit concevoir un modèle fonctionnel des différentes connaissances recueillies.

2.1. En théorie

La méthode KADS permet donc de traiter tout le processus d'acquisition des connaissances, du recueil de celles-ci dans la phase d'analyse au développement d'un système complet dans la phase de conception. Durant la phase d'analyse, le novice analyse la situation réelle à l'aide d'un outil d'interprétation, ce qui lui permet d'aboutir à un modèle conceptuel qui sera rendu fonctionnel durant la phase de conception. Cet outil d'interprétation est dirigé par une bibliothèque de six modèles. L'explication de ces différents modèles se base sur le livre de Prax. [PRAX00]

2.1.1. Le modèle d'expertise

Le modèle d'expertise est le modèle principal qui tente de modéliser les processus par lesquels un expert trouve une solution à un problème. L'expert est censé suivre des règles et une stratégie spécifique lors de son travail. Ces règles et cette stratégie sont tacites ou explicites. Le novice essaie donc de formaliser les règles et la stratégie de l'expert en se basant premièrement sur la connaissance du domaine, c'est-à-dire les concepts théoriques et les relations entre les différents concepts. Deuxièmement, il utilise les structures d'inférences, c'est-à-dire les règles appliquées par l'expert pour manipuler ces différents concepts. Troisièmement, le novice exploite la connaissance des tâches, c'est-à-dire le contexte dans lequel l'expert travaille. Quatrièmement, il analyse la stratégie de résolution de problèmes déployée par l'expert.

2.1.2. Le modèle organisation

Le modèle organisation décrit l'entreprise dans laquelle le système sera exploité. Ce modèle permet d'identifier les zones prometteuses de l'entreprise pour un système basé sur la connaissance et analyse l'impact de celui-ci sur l'activité.

2.1.3. Le modèle tâche

Le modèle tâche décrit les différentes tâches qui sont exécutées dans l'environnement où l'on se propose d'installer le système basé sur la connaissance. Les tâches sont décrites indépendamment des agents qui les exécutent.

2.1.4. Le modèle agent

Le modèle agent décrit les attributions des agents qui exécutent les tâches. Il identifie tous les agents qui sont concernés par le projet et les futurs utilisateurs du système. Ces agents peuvent être soit des personnes, soit des applications informatiques.

2.1.5. Le modèle communication

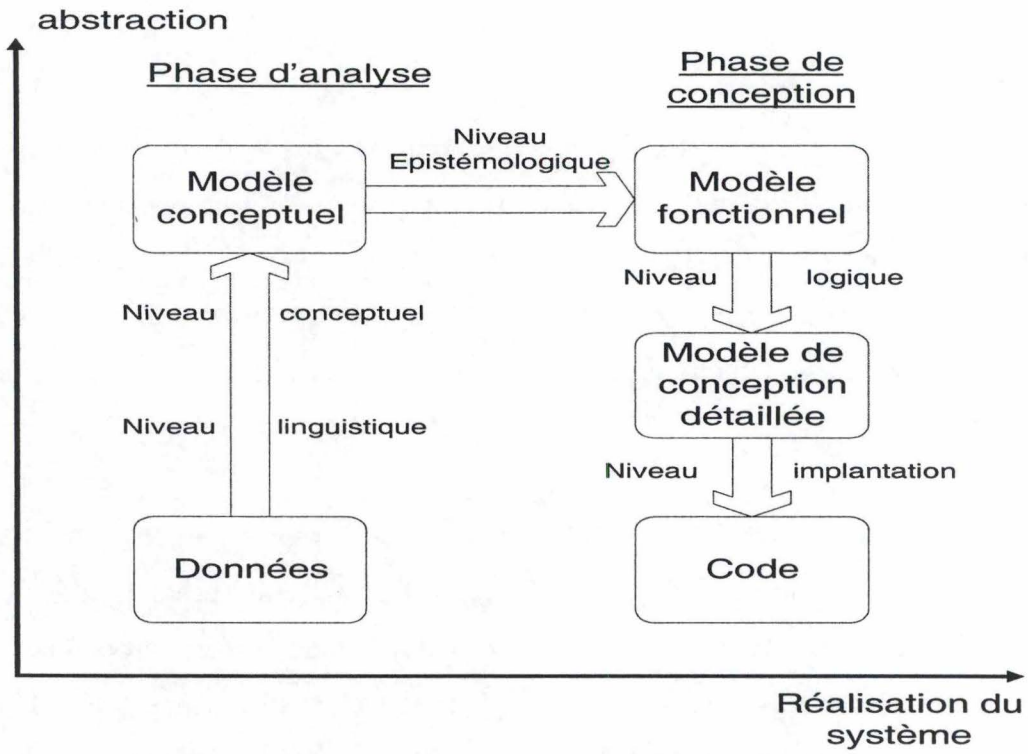
Le modèle communication décrit les interactions entre les différents agents du système et fournit les mécanismes nécessaires pour améliorer les capacités communicantes de ces agents lors des exécutions des tâches.

2.1.6. Le modèle de conception

Le modèle de conception est le lien entre les différents modèles conceptuels (expertise, organisation, tâche, agent et communication) ainsi qu'entre leur implémentation informatique. Il s'attache donc à l'architecture et à la fonctionnalité détaillée du système.

2.2. En pratique

La méthode KADS suit une démarche descendante. Grâce à sa bibliothèque de modèles, elle présente rapidement une modélisation de l'expertise. La modélisation suit le schéma suivant. (SCHEMA 2.1)



SCHEMA 2.1 : Modèles et espace de développement de KADS.

L'intérêt de cette méthode se base lui aussi sur la conception de modèles conceptuels. Ces modèles conceptuels sont dirigés par une bibliothèque de six modèles. En plus, il existe une réelle distinction entre la phase d'analyse et la phase de conception. L'expert n'aura donc plus rien à dire une fois qu'il aura approuvé la phase d'analyse. La méthode KADS est donc fortement dirigée.

3. REX

La méthode REX (Retour d'EXpérience) a été élaborée en 1993 par le français Prieur et a pour objectif la capitalisation des connaissances ainsi que la formalisation des retours d'expérience. Cette méthode a été initiée et développée par le Commissariat à l'Energie Atomique (CEA) dans le but de préserver les savoirs et savoir-faire acquis pendant les phases de conception et de mise en route des réacteurs nucléaires. Cette méthode se fonde sur une méthodologie assistant le processus de formalisation de l'expérience et un outil logiciel de gestion de cette expérience ainsi formalisée. Cet outil logiciel est développé par la firme Euriware.

3.1. En théorie

Cette méthode repose sur trois phases : la première est l'analyse des besoins et l'identification des sources de connaissance d'une organisation ; la seconde reprend la construction d'Eléments de Connaissance (EC) grâce à des interviews réalisées auprès des experts de cette organisation, aux documents de ceux-ci et à la consultation de bases de données déjà existantes ; la dernière consiste en la construction d'un système de gestion des connaissances.

Qu'entendons-nous par EC et sous quelle forme est-il représenté ? Un EC est un texte qui a pour but de valoriser les connaissances de l'organisation et de faciliter leur consultation. Un EC est représenté sous la forme d'une fiche structurée en trois blocs de texte. Tout d'abord, **la partie contextuelle** fournit le contexte dans lequel s'est déroulé ou est apparu l'EC. Ensuite, **la partie factuelle** donne un résumé de la connaissance ainsi que la description d'un fait observé ou d'une pratique. Finalement, **la partie analytique** permet d'identifier le point de vue de l'auteur concernant cet EC.

Ainsi, si nous nous trouvons dans une centrale nucléaire et que se présente une fuite dans un des réacteurs, nous aurons dans la partie contextuelle de la fiche intitulée 'réparer une fuite de réacteur' : apparition d'une fuite dans un réacteur de type A suite à l'utilisation du processus XYZ. Dans la partie factuelle, nous retrouverons : fermer la vanne, mettre en route le système d'alarme, évacuer la section, irradier la section. Dans la partie analytique, nous aurons : il serait préférable de donner l'alarme en premier lieu, il faut vérifier que plus personne ne se trouve dans la section.

Il existe trois types différents d'EC. Les Eléments de Connaissance Documentaires (**ECD**) correspondent au résumé d'un document. Les Eléments d'EXpérience (**EEX**) représentent l'expérience acquise d'un employé et se formalisent au cours des entretiens. Les Eléments de Savoir-Faire (**ESF**) correspondent au savoir-faire acquis par un employé en participant à une activité particulière.

3.2. En pratique

La mise en place du système de gestion des connaissances doit permettre à un individu d'interroger le système en langage libre. Pour ce faire, le système utilise une recherche par mots clés. Suite à l'introduction des mots clés choisis pour la connaissance recherchée, le système doit renvoyer à l'utilisateur les EC sous forme de dossiers ordonnés par ordre décroissant de pertinence par rapport à la requête. Pour cela la méthode REX gère les fiches de connaissance par un système de gestion de base de données orientée objet et fonctionne dans un environnement Intranet, ce qui autorise une consultation des fiches depuis un simple navigateur.

Selon Euriware, une filiale du CEA, le progiciel REX a pour but de mettre à la disposition de ceux qui pourraient en avoir besoin toutes les connaissances accumulées lors de la réalisation d'un projet dans une base de données. Les auteurs de ces bases de données s'interdisent d'éliminer les doublons. La même connaissance peut être présentée plusieurs fois, selon plusieurs points de vue : celui du concepteur, celui du responsable de la maintenance, celui de l'utilisateur. Cela permet ainsi de décloisonner les métiers. [DESS SID97]

L'intérêt de cette méthode est la présentation des connaissances sous forme de fiches qui sont structurées et claires. Elles sont faciles à lire et à comprendre et elles sont dotées de liens vers d'autres fiches (si celles-ci existent) qui peuvent, si nécessaire, donner un complément d'informations. Malheureusement, la méthode est assez lourde car elle suppose trois interviews par employé.

4. MKSM

La méthode MKSM (Method for Knowledge System Management) réalisée en 1993 par Ermine et son équipe a pour but d'élaborer une méthode opérationnelle permettant de maîtriser la complexité d'un système de connaissances pour réaliser des objectifs opérationnels. Cette méthode a été élaborée au sein du 'Groupe Gestion des Connaissances' de la direction de l'Information Scientifique et Technique du CEA.

Selon Ermine, la méthode MKSM comporte plusieurs objectifs : « Développer des fondements théoriques importants et solides. (...) Fournir un ensemble de méthodes et d'outils directement appropriables par des utilisateurs 'néophytes', avec un minimum d'effort et de connaissances spécialisées. (...) Etre une 'méthode brève'. (...) Viser un large choix d'applications, car la gestion des connaissances recoupe un grand nombre d'autres projets. (...) Se baser sur des expériences concrètes et variées. (...) Permettre une évaluation à terme d'un retour d'investissement possible. (...) MKSM se présente donc comme une méthode d'analyse des systèmes de connaissances, et a pour but de rendre ces systèmes intelligibles à ceux qui en sont les acteurs, afin qu'ils puissent mettre eux-mêmes en place leur propre système de connaissances. ». [ERMINE96]

Pour répondre à une partie de ces objectifs, il faut dès lors que la méthode MKSM puisse être utilisée par des outils informatiques de style bureautique tels que Word, Excel, Visio ; c'est-à-dire, des outils logiciels qui s'intègrent aisément dans l'univers classique des employés de l'organisation.

4.1. En théorie

La méthode MKSM procède par une série de modélisations de plus en plus fines qui se réalisent suivant certaines phases. Selon la nature du problème, toutes les phases ne seront pas nécessairement appliquées. En pratique, la méthode MKSM comporte cinq phases de modélisation que l'on appelle 'cycle de modélisation'. Nous allons maintenant décrire ces différentes phases.

4.1.1. Le modèle du système de référence

Le modèle du système de référence consiste en la définition précise du système dont le patrimoine de connaissances est à gérer. On identifie les différents acteurs ainsi que leur patrimoine de connaissances respectif. Par le terme d'acteurs, on entend bien entendu les personnes physiques mais aussi le matériel. Tout ce patrimoine rentre alors dans le système d'informations proprement dit.

4.1.2. Le modèle du domaine

Dans le modèle du domaine, il s'agit de décrire les connaissances en contexte, c'est-à-dire de se poser la question : de quoi parle la connaissance ? Pour y répondre, nous construisons le modèle du domaine dans lequel nous reprenons tous les processus de l'organisation. On attache à chaque élément du patrimoine de connaissances du modèle un ensemble de propriétés qui prennent souvent la forme de grandeurs qui servent à caractériser ces éléments.

4.1.3. Le modèle d'activité

Dans le modèle d'activité, nous nous posons la question suivante : dans quelle activité la connaissance est-elle mise en œuvre ? On procède pour cela à une analyse de type 'fonctionnelle' descendante, où chaque activité est décomposée hiérarchiquement en sous activités de plus bas niveau. La modélisation décrit l'activité en termes d'entrées, de sorties, de ressources et d'acteurs.

4.1.4. Le modèle des concepts

Le modèle des concepts modélise les connaissances statiques, c'est-à-dire une description sémantique des objets, des concepts et des attributs du domaine. Un concept désigne une catégorie de connaissances ayant des propriétés communes. Ainsi, une connaissance appartient à un concept, elle est une instance de concept. Un concept possède des attributs. Il existe aussi une relation de spécialisation et d'héritage entre les différents concepts.

4.1.5. Le modèle des tâches

Le modèle des tâches modélise les connaissances dynamiques, c'est-à-dire qu'il fournit une représentation de la stratégie de résolution de problèmes ou du processus d'utilisation des connaissances statiques. La méthode MKSM construit dès lors une

arborescence qui affine récursivement les tâches de plus haut niveau en sous tâches plus détaillées, jusqu'à aboutir à des tâches dites terminales.

4.2. En pratique

Les premiers résultats obtenus sont constitués par l'ensemble de modèles vu précédemment formalisant la connaissance recueillie lors des interviews. Ces modèles sont complétés par des fiches descriptives synthétiques, puis par des synthèses complémentaires rédigées par les experts. Le résultat final est appelé 'le Livre de Connaissances'. C'est donc un livre reprenant toutes les connaissances qui peut se présenter sous un format papier ou électronique.

L'intérêt de cette méthode se retrouve dans le fait qu'elle est supportée par des outils simples. Cette méthode se réalise suivant un degré de précision de plus en plus fin à chaque étape. Il y a donc un avancement préétabli qui permet une modélisation précise.

III. Fiches d'identité

Nous allons dans cette dernière partie établir ce que l'on pourrait appeler les fiches signalétiques des quatre méthodes construites vues précédemment. Ces fiches nous résumeront en quelques points ces différentes méthodes. Il est important de noter que ces méthodes ne sont pas concurrentes. Elles doivent plutôt être vues comme complémentaires dans la mesure où elles s'orientent vers des finalités différentes.

Nom :	MACAO (Méthode d'Acquisition des Connaissances Assistée par Ordinateur).
Auteur(s) :	Nathalie Aussenac-Gilles (projet de l'IRIT).
Objectifs de la méthode :	Fournir aux novices un support méthodologique et logiciel à l'acquisition des connaissances.
Première application de la méthode :	EDF-DER, Matra-Marconi Space, ARAMIIHS (milieu industriel). Projet Sisyphus, collaboration avec LRI (milieu universitaire).
Date d'élaboration :	1988.
Pays d'origine :	France.
Description de la méthode :	Construction d'un modèle conceptuel décrivant l'expertise nécessaire pour réaliser la tâche assignée au futur système. Ce modèle sert de support au dialogue entre novice et expert.
Outils logiciels :	MONA assisté de LISA.
Avantages :	MACAO est un bon support pour la phase de modélisation des connaissances.
Faiblesses :	Les modèles construits ne sont pas directement opérationnels. Ils sont spécifiques à chaque expertise et donc non réutilisables.
Intérêts :	Modélisation dans un modèle conceptuel des connaissances du domaine ou du raisonnement de l'expert.

FICHE 2.1 : Fiche d'identité de MACAO.

Nom :	KADS (Knowledge Analysis and Design System).
Auteur(s) :	Anne Brooking, Joost Breuker, Bob Wielinga et Mike Rogers.
Objectifs de la méthode :	Aider à la modélisation des connaissances d'un expert ou d'un groupe d'experts dans le but de réaliser un système d'aide à la décision basé sur les connaissances.
Première application de la méthode :	EDF.
Date d'élaboration :	1989 pour KADS et 1992 pour KADS-II.
Pays d'origine :	Europe.
Description de la méthode :	Modélisation conceptuelle des connaissances en plusieurs étapes successives allant du général au particulier, passant du modèle d'expertise au modèle de conception.
Outils logiciels :	Outils propres à la méthode : KADS tools, OpenKADS.
Avantages :	Méthode qui s'adapte à toute situation d'expertise.
Faiblesses :	Outils logiciels lourds.
Intérêts :	Direction par une bibliothèque de six modèles. Distinction entre la phase d'analyse et la phase de conception.

FICHE 2.2 : Fiche d'identité de KADS.

Nom :	REX (Retour d'Expérience).
Auteur(s) :	Patrick Prieur.
Objectifs de la méthode :	Capitaliser les connaissances et favoriser le retour d'expérience.
Première application de la méthode :	Projet Accore au CEA (Accès aux connaissances relatives aux réacteurs nucléaires).
Date d'élaboration :	1993.
Pays d'origine :	France.
Description de la méthode :	Constitution d'une base de connaissances à partir d'expériences humaines, mémorisation et valorisation de la mémoire par réintroduction des savoirs au niveau du savoir-faire individuel.
Outils logiciels :	Progiciel REX.
Avantages :	Méthode faisant l'objet d'une large diffusion.
Faiblesses :	Méthode très lourde à mettre en œuvre.
Intérêts :	Présentation des connaissances sous forme de fiches qui sont structurées et claires.

FICHE 2.3 : Fiche d'identité de REX.

Nom :	MKSM (Method for Knowledge System Management).
Auteur(s) :	Jean-Louis Ermine, Mathias Chaillot, Philippe Bigeon, Boris Charreton, Denis Malaveille.
Objectifs de la méthode :	Maîtriser la complexité dans les projets de gestion des connaissances, avant d'aboutir à un projet 'opérationnel'.
Première application de la méthode :	CEA (Commissariat à l'Energie Atomique).
Date d'élaboration :	1993.
Pays d'origine :	France.
Description de la méthode :	La méthodologie procède par entretien pour explorer le système cognitif de chaque expert. Elle aboutit à un classement des récits obtenus en différents types de connaissances. Les phases de la méthode procèdent par affinages successifs de la modélisation du patrimoine de connaissances.
Outils logiciels :	Utilisation de logiciels bureautiques comme Word, Excel.
Avantages :	Existence d'un large champ d'application. Le principe de modélisation a un grand pouvoir de représentation et de communication entre les acteurs.
Faiblesses :	La modélisation peut parfois s'avérer lourde à mettre en place.
Intérêts :	Simplicité des outils. Degré de précision de plus en plus fin à chaque étape.

FICHE 2.4 : Fiche d'identité de MKSM.

Conclusion de chapitre

Nous pouvons dire que nous avons dans ce second chapitre fait un tour d'horizon des différentes méthodes existantes pour la gestion des connaissances. Nous avons insisté sur le fait que ces différentes méthodes ne sont pas des outils de gestion des connaissances mais bien des méthodes même si certaines sont complétées par des outils spécifiques. Nous n'avons pas décrit toutes les méthodes de gestion des connaissances mais celles les plus fréquemment énoncées par les différents auteurs.

Nous avons abordé dans ce chapitre la méthode MACAO qui permet d'extraire les connaissances et de les organiser avec l'aide de l'expert ; la méthode KADS qui aide à la modélisation des connaissances d'un expert ou d'un groupe d'experts dans le but de réaliser un système d'aide à la décision basé sur la connaissance ; la méthode REX qui donne la possibilité de capitaliser les connaissances et de favoriser le retour d'expérience ainsi que la méthode MKSM qui offre de maîtriser la complexité dans les projets de gestion des connaissances avant d'aboutir à un projet opérationnel. Toutes ces méthodes ne sont pas concurrentes. Elles doivent plutôt être vues comme complémentaires dans la mesure où elles s'orientent vers des finalités différentes.

Ce chapitre clôt la première partie de notre présent travail concernant les pré-requis théoriques en matière de gestion des connaissances. Nous allons donc maintenant aborder la seconde partie qui sera beaucoup plus pratique car nous allons y retrouver le prototype d'outil de gestion des connaissances que nous avons en partie développé.

Seconde partie

Création d'un prototype de gestion des connaissances

Troisième chapitre : Généralités sur le prototype Anthémis

Quatrième chapitre : Technologies utilisées dans le prototype Anthémis

Cinquième chapitre : Langages utilisés dans le prototype Anthémis

Sixième chapitre : Modèle Entité-Association pour le prototype Anthémis

Septième chapitre : Le formalisme des graphes conceptuels pour gérer la connaissance

Huitième chapitre : Evaluation du prototype Anthémis

Troisième chapitre

Généralités sur le prototype Anthémis

I. Présentation du contexte de travail

1. Pré-requis nécessaires à l'élaboration du prototype
2. Objectifs du prototype Anthémis

   2.1. Objectifs généraux
      2.1.1. Insertion des connaissances
      2.1.2. Gestion des connaissances
      2.1.3. Présentation des connaissances
   2.2. Objectifs de développement

3. Présentation des deux approches du prototype Anthémis

   3.1. Approche basée sur le modèle Entité-Association
   3.2. Approche basée sur les graphes conceptuels

II. Présentation du prototype Anthémis

1. Découpe en modules

   1.1. Description des modules
   1.2. Interactions entre les modules

2. Fonctionnalités du prototype Anthémis

   2.1. Collecte des connaissances
      2.1.1. Page de chargement
   2.2. Production de connaissances
      2.2.1. Pages de création, de modification et de suppression de connaissances
   2.3. Diffusion des connaissances
      2.3.1. Pages sur les listes des entités, des concepts et des relations
      2.3.2. Méthode de recherche dans l'approche graphes conceptuels

3. Interface Homme-Logiciel
4. Fiche d'identité du prototype

Introduction de chapitre

Le but de notre stage consistait au développement d'un prototype permettant de gérer les connaissances. Ce prototype part d'une idée originale du Professeur Gerbé qui souhaite créer un logiciel unique sur le marché. Comme nous le verrons, la grande nouveauté de ce prototype réside surtout dans la manière de stocker les connaissances dans la base de données ainsi que dans la manière de présenter celles-ci. La création de ce prototype s'est réalisée suivant deux approches que nous détaillerons.

Nous analyserons dans ce chapitre le contexte de travail dans lequel nous étions plongé. Ainsi, nous parlerons de l'ensemble des pré-requis que nous avons dû acquérir pour pouvoir travailler sur le prototype, que ce soit des langages de programmation, des technologies ou encore le formalisme des graphes conceptuels. Nous préciserons aussi les objectifs que veut atteindre le Professeur Gerbé avec le prototype Anthémis.

Ensuite, nous introduirons brièvement les deux approches utilisées pour la création du prototype. En effet, le prototype a été développé sous deux approches, la première étant plus classique, la deuxième plus expérimentale. Le prototype se base sur une conceptualisation des connaissances soit sous forme de modèle Entité-Association, soit sous forme de graphes conceptuels. Nous ne rentrerons pas, ici, dans les détails car ces deux approches seront présentées dans le sixième et le septième chapitre.

Finalement, nous exposerons les principes généraux du prototype Anthémis, nous étudierons ses différentes fonctionnalités. Nous détaillerons également une découpe en modules de son implémentation et nous terminerons par la présentation de sa fiche d'identité.

I. Présentation du contexte de travail

Notre stage s'est effectué aux Hautes Etudes Commerciales (HEC) de Montréal. Nous nous trouvions au sein du département technologies de l'information sous la supervision du Professeur Gerbé. Nous avons travaillé à l'élaboration d'un prototype, dénommé Anthémis, qui doit contribuer à gérer efficacement les connaissances au sein d'une organisation et qui provient d'une idée originale du Professeur Gerbé.

1. Pré-requis nécessaires à l'élaboration du prototype

Avant de pouvoir travailler correctement sur le prototype Anthémis, nous avons dû acquérir plusieurs pré-requis concernant celui-ci.

Tout d'abord, nous avons dû nous plonger dans le travail réalisé par les deux étudiants qui nous avaient précédés. Ces deux étudiants avaient déjà commencé à élaborer ce que nous allons appeler le noyau du prototype, c'est-à-dire la création de la base de données et la connexion de différentes procédures manuelles de traitement des données. Le but de notre travail était, quant à lui, de reprendre le noyau que nos prédécesseurs avaient créé et de l'étendre en permettant l'introduction des connaissances et de leurs modèles par des fichiers XML ainsi qu'en permettant la gestion des connaissances par une interface et non plus de manière manuelle. Ainsi, notre premier pré-requis était de reprendre l'élaboration du prototype en cours, de comprendre ce qui avait été fait et de continuer dans la même voie.

Ensuite, le formalisme des graphes conceptuels nous étant totalement inconnu, nous avons dû nous plonger dans sa théorie pour en comprendre le fonctionnement. Nous reprendrons dans ce chapitre les concepts de base qui permettront de donner une idée de la représentation des graphes conceptuels et nous les détaillerons plus précisément, dans le septième chapitre, en nous basant sur la thèse du Professeur Gerbé intitulée *'Un modèle uniforme pour la modélisation et la métamodélisation d'une mémoire d'entreprise'*. [GERBE00-1]

Par après, l'élaboration du prototype requiert l'utilisation de différentes technologies plus ou moins complexes. Le choix de ces dernières ayant été effectué par les étudiants précédents et par le Professeur Gerbé, nous avons dû nous y plier. Or, ces technologies, pour la plupart d'entre elles, nous étaient inconnues. Nous avons dû nous familiariser avec elles dès notre arrivée. Nous avons donc appris à maîtriser Personnal Web Server de Microsoft et la technologie ASP. C'est pour cette raison que nous les présenterons dans le quatrième chapitre.

Finalement, nous avons dû apprendre différents langages de programmation choisis par nos prédécesseurs et le Professeur Gerbé. Ce fut le cas notamment des langages XML, HTML, XSL et JScript. Nous reprendrons ces différents langages dans le cinquième chapitre.

2. Objectifs du prototype Anthémis

2.1. Objectifs généraux

L'objectif poursuivi est donc la création d'un logiciel capable de gérer les connaissances. Ce logiciel doit être accessible à tous les employés d'une organisation, ce qui nous a contraints ainsi à limiter les pré-requis nécessaires à son utilisation. Nous allons maintenant reprendre les trois objectifs principaux du prototype Anthémis.

2.1.1. Insertion des connaissances

Ce logiciel doit laisser à l'utilisateur la possibilité d'insérer lui-même ses connaissances. Mais surtout, ce logiciel ne doit pas être limité à un modèle unique qui définit une fois pour toute la structure générale que doivent suivre les connaissances mais doit permettre d'insérer plusieurs modèles. En effet, l'idée innovante du logiciel est de permettre à l'utilisateur de créer son propre modèle pour ses propres connaissances et de ce fait permettre de stocker dans la base de données un éventail très large de connaissances différentes. En réalité, l'utilisateur devra avant tout insérer dans la base de données un

modèle qui définira la structure que prendront ses connaissances et ensuite il pourra insérer les connaissances liées à ce modèle. Ainsi, la base de données sera capable de gérer aussi bien des connaissances définissant un processus de travail que des connaissances expliquant la hiérarchie de pouvoir se trouvant dans l'organisation et ne sera plus limitée au modèle prédéfini par les créateurs du logiciel.

2.1.2. Gestion des connaissances

Le deuxième objectif du prototype Anthémis consiste à garantir aux utilisateurs de pouvoir gérer leurs connaissances insérées dans la bases de données. La gestion des connaissances doit permettre à l'utilisateur de réaliser plusieurs manipulations sur celles-ci. La première manipulation consiste à créer de nouvelles connaissances dans un modèle défini, la seconde à modifier des connaissances et la dernière à supprimer des connaissances.

2.1.3. Présentation des connaissances

Le troisième objectif très important se retrouve dans la capacité du logiciel à créer des pages HTML uniformes pour la visualisation des connaissances malgré le fait que ces dernières peuvent suivre des modèles complètement différents. Effectivement, la présentation des connaissances suit un métamodèle qui permet de les présenter suivant des pages HTML standardisées et suivant toujours la même structure. Nous reparlerons de cette présentation ultérieurement dans le présent chapitre.

2.2. Objectifs de développement

L'idée principale du Professeur Gerbé est de créer le prototype en deux temps. D'une part, une partie du prototype est centrée sur une approche Entité-Association. Cette approche est plus conventionnelle et se rapproche plus de ce qui existe sur le marché. D'autre part, dans sa thèse précédemment citée, le Professeur Gerbé propose un métamodèle et un langage pour la représentation des connaissances basés sur les graphes

conceptuels. Cette deuxième approche se trouve encore au stade expérimental et viendra, dans l'avenir, se greffer sur la première afin de rendre le logiciel plus puissant.

Le travail effectué sur le prototype avait, dès lors, deux buts principaux. Premièrement, le prototype basé sur le modèle Entité-Association a pour but de définir les grandes lignes de développement et d'agencement des fonctionnalités dans l'optique d'une commercialisation rapide. Deuxièmement, l'approche selon les graphes conceptuels réside dans une validation de la théorie du Professeur Gerbé. Le prototype suivant l'approche des graphes conceptuels est ainsi développé pour tester la puissance de ces graphes afin de gérer une base de connaissances et aussi pour vérifier la puissance de ces graphes par rapport au modèle Entité-Association.

3. Présentation des deux approches du prototype Anthémis

Nous allons ici présenter les deux approches utilisées par le prototype Anthémis. Ces deux approches représentent des formalismes différents de modélisation de données dans une base de données. Nous commencerons par examiner l'approche basée sur le modèle Entité-Association et nous terminerons par celle basée sur les graphes conceptuels. Nous ne nous attarderons pas sur l'explication de ces deux approches car nous les verrons en détails dans le sixième et le septième chapitre.

3.1. Approche basée sur le modèle Entité-Association

Le modèle Entité-Association permet de représenter naturellement les différents concepts que nous souhaitons formaliser dans une base de données sans s'attacher à tous les aspects techniques liés à l'introduction de ces concepts dans la base de données elle-même. Cette représentation se réalise sous forme graphique qui est simple à comprendre et attrayante pour l'utilisateur.

Le modèle Entité-Association est perçu selon le Professeur Hainaut « sous la forme d'ensembles d'entités. Les entités sont en association les unes avec les autres. Les

entités d'une classe sont du même type. Elles sont caractérisées par des attributs qui décrivent leurs propriétés intrinsèques. Les types d'associations sont caractérisés par leur classe, qui indique combien d'entités d'un type sont associées à une entité de l'autre type, et par leur caractère obligatoire ou facultatif pour les entités associées. Un type d'entités a un identifiant, généralement constitué d'attributs. ». [HAINAUT94, p73]

Nous parlerons explicitement de l'approche basée sur le modèle Entité-Association dans le sixième chapitre du présent travail car nous avons travaillé sur le prototype en suivant cette approche.

3.2. Approche basée sur les graphes conceptuels

Nous allons décrire brièvement dans cette section le fonctionnement des graphes conceptuels. Nous verrons également l'intérêt de suivre une telle approche.

La théorie des graphes conceptuels est un nouveau formalisme qui a vu le jour en 1984. Nous devons le développement de cette théorie à Sowa qui l'a introduite dans son livre *'Conceptual Structures : Information Processing in Mind and Machine'*. [SOWA84] Ce formalisme très simple se rapproche fortement de la langue naturelle. Il est compris aussi bien par l'homme que par les machines. De plus, il facilite l'échange de données via le protocole Conceptual Graph Interchange Form (CGIF). Les graphes contiennent deux types de notion qui sont les concepts et les relations. Les concepts se représentent par des rectangles et les relations par des cercles. Des arcs joignent ces deux notions pour compléter les graphes. Nous présenterons dans le septième chapitre une introduction plus détaillée aux graphes conceptuels et nous verrons l'intérêt de choisir ce formalisme pour gérer les bases de connaissances.

L'approche graphes conceptuels pour le prototype est avant tout une approche plus expérimentale. En effet, dans sa thèse intitulée *'Un modèle uniforme pour la modélisation et la métamodélisation d'une mémoire d'entreprise'* [GERBE00-1], le Professeur Gerbé présente un langage de représentation des connaissances par le formalisme des graphes conceptuels. Le prototype avait donc pour but de valider cette

théorie et de montrer la puissance de ce formalisme par rapport à d'autres approches et notamment celle selon le modèle Entité-Association.

Nous exposerons donc cette approche dans le septième chapitre et nous analyserons dans le huitième chapitre les différences majeures qui existent entre ces deux approches. La plus grande étant la présence d'un outil de recherche puissant présenté par après dans ce chapitre.

II. Présentation du prototype Anthémis

Nous allons, pour commencer cette partie, exposer les différents modules qui constituent notre prototype. Nous décrirons tous ces modules ainsi que l'interaction de ceux-ci. Cette découpe en modules permettra de donner une bonne vision d'ensemble du prototype.

Une fois les différents modules exposés, nous allons dégager les différentes fonctionnalités du prototype Anthémis. Ces fonctionnalités vont être illustrées par des captures d'écrans de notre prototype que ce soit pour l'approche Entité-Association ou celle des graphes conceptuels. Les fonctionnalités seront présentées suivant les trois buts de la gestion des connaissances.

1. Découpe en modules

Pour avoir une meilleure vision d'ensemble du prototype, nous allons présenter dans cette section une découpe du prototype en modules. Celui-ci contient quatre modules : le module base de données, le module interface graphique de gestion, le module présentation et le module auxiliaire. Pour mieux comprendre ces différents modules, nous allons d'abord en décrire le contenu et ensuite nous montrerons leurs interactions.

1.1. Description des modules[1]

Le module base de données contient tous les fichiers modifiant la base de données. Nous retrouvons donc ici le fichier initBD.asp pour l'approche Entité-Association et createMetabase.asp pour l'approche graphes conceptuels permettant de vider les tables de la base de données lors de l'initialisation de celle-ci. Ensuite, nous avons tous les fichiers qui permettent d'insérer les modèles et les connaissances dans la base, c'est le cas entre autre des fichiers creermodel.asp ou creerdonnee.asp ainsi que preCreateData.asp ou createData.asp pour l'approche graphes conceptuels. Et enfin, nous retrouvons tous les fichiers qui communiquent avec la base de données pour la modifier, comme le fichier MAJ.asp et modif.asp pour les graphes conceptuels.

Le module interface graphique de gestion contient tous les fichiers qui permettent à l'utilisateur de naviguer dans les différentes pages, pages permettant la gestion des diverses connaissances. Nous retrouvons dans ce module les différents fichiers affichant des informations à l'écran, c'est le cas des fichiers chargement.html, ListType.asp, ListObjet.asp ou main.html, listeData.asp et presentation.asp pour les graphes conceptuels. Nous retrouvons aussi les fichiers affichant les formulaires concernant la modification, la création ou la suppression de connaissances comme les fichiers créer.asp, suppr.asp, lien.asp ou edit.asp ainsi que modifData.asp et createData.asp pour les graphes conceptuels.

Le module présentation contient l'ensemble des fichiers permettant la diffusion et la présentation des connaissances. Ceux-ci permettent d'obtenir une présentation uniforme et standardisée des connaissances. La page de présentation créée par ces différents fichiers génère des hyperliens garantissant une navigation entre les pages concernant la connaissance présentée. Les fichiers de ce module ont été créés par le Professeur Gerbé et ne sont pas à notre disposition, nous ne saurons donc pas les présenter dans ce présent travail.

Ce que nous pouvons affirmer pour la présentation, c'est que les connaissances seront affichées dans des pages divisées en trois sections : la première, que nous pouvons appeler **l'énonciation**, reprend le code de la connaissance visionnée par l'utilisateur ; la

[1] Tous les fichiers cités dans cette section sont présentés dans les ANNEXES 1 et 5 du présent travail.

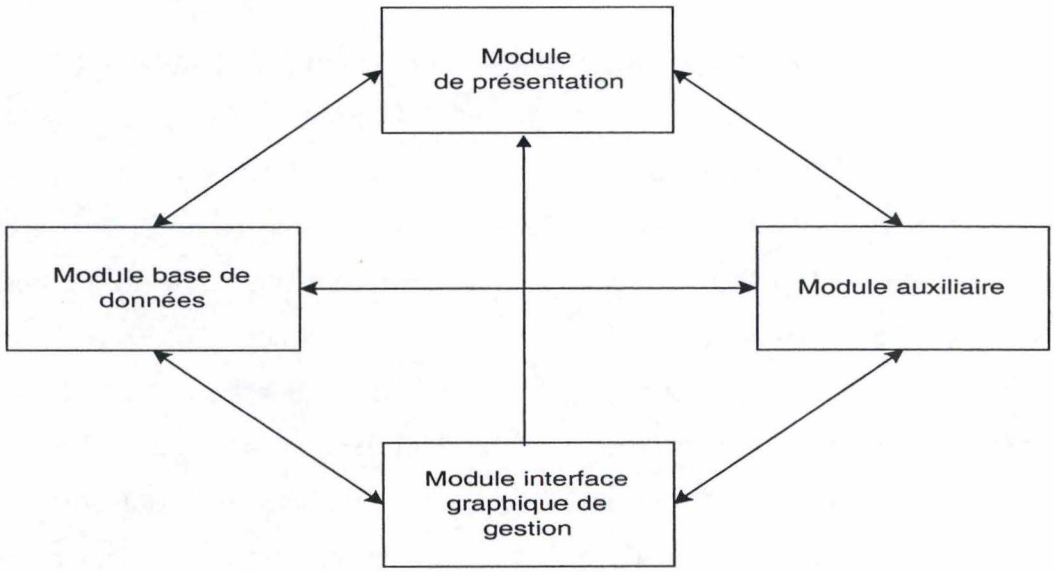
deuxième, que nous appellerons **la description**, donne la description de cette connaissance en reprenant les différentes valeurs trouvées dans la base de données pour l'objet concernant cette connaissance ; la troisième, que nous dénommerons **les liens**, affiche les différents liens que cette connaissance entretient avec d'autres connaissances de la base de données. Il sera donc possible de cliquer sur ces liens pour visionner les autres connaissances en correspondance.

Illustrons ces propos par un cas concret. Supposons qu'il existe comme connaissance plusieurs projets différents. L'énonciation correspondra à la liste de tous ces différents projets : Anthemis, Ariane, etc. La description permet de présenter les caractéristiques de chaque projet et ses relations : Ariane est un projet de conquête spatiale mené par l'Europe, ... Enfin, les liens permettront de naviguer d'une page à l'autre. Si nous reprenons le cas du projet Anthemis, en cliquant simplement sur Europe, nous aurons accès à une nouvelle page nous donnant les caractéristiques de l'Europe, comme ses pays membres, etc.

Nous ne pouvons donner plus d'explications sur le fonctionnement pratique de la génération de ces pages uniformes car le module présentation ne rentrait pas dans notre domaine de travail durant le stage réalisé à Montréal et ce module est en cours de développement par le Professeur Gerbé.

Le module auxiliaire contient quant à lui un ensemble de fonctions utilisées par les autres modules. Ces fonctions dites génériques contiennent entre autres les variables globales du logiciel appelables par tous les modules. Le module contient aussi les fonctions de connexion et des fonctions génériques permettant d'exécuter les requêtes SQL automatiquement.

1.2. Interactions entre les modules



SCHEMA 3.1 : Interactions des différents modules.

Sur le SCHEMA 3.1, nous pouvons remarquer que le module auxiliaire est appelé naturellement par les trois autres modules et celui-ci renvoie des résultats à ces mêmes modules. En effet, le module auxiliaire étant constitué de fonctions génériques, il contient certaines fonctions utilisées par les trois autres modules. Celles-ci sont donc logiquement appelées par les trois autres modules et en retour, le module auxiliaire leur renvoie le résultat.

Le module base de données peut être, quant à lui, appelé par le module de présentation ainsi que par le module interface graphique de gestion pour communiquer avec la base de données. En effet, le module base de données contient l'ensemble des fonctions nécessaires pour l'accès à la base de données. Ainsi, le module présentation et le module interface graphique de gestion appellent ces fonctions via le module base de données et en retour, le module base de données leur renvoie les résultats.

Le module interface graphique de gestion réalise un simple appel au module de présentation lorsqu'un utilisateur veut visionner une connaissance. Il n'existe pas par ailleurs d'appel dans l'autre sens. Ces deux modules diffèrent l'un de l'autre car le module présentation ne contient que les fonctions pour la présentation finale de la connaissance

alors que le module interface graphique de gestion comporte les différentes pages permettant la navigation dans le prototype.

Il faut insister, dès à présent, sur le fait que notre travail ne s'est pas effectué sur l'ensemble des modules. En effet, nous avons principalement travaillé sur le module base de données et sur le module interface graphique de gestion, le module de présentation étant laissé aux bons soins du Professeur Gerbé. Quant au module auxiliaire, celui-ci faisant partie du noyau, nous y avons juste inséré un ensemble de fonctions et de méthodes génériques nous aidant dans notre tâche. Rappelons aussi que notre stage consistait à travailler sur un prototype, nous avons donc contribué à mettre en place les idées que le Professeur Gerbé avaient. Notre but n'était donc pas d'arriver à la fin de notre stage à un logiciel prêt à être commercialisé mais plus à tester les différentes idées du Professeur Gerbé. Notre travail est donc un prototype qui comporte encore des zones d'ombres qui devront être éclaircies.

2. Fonctionnalités du prototype Anthémis

Comme annoncé, le premier objectif du prototype Anthémis est de permettre une modélisation cohérente et structurée des connaissances dans une base de données. Or de nos jours, les bases de données présentent le défaut d'être soit trop structurées, dans le cas d'une base de données où un modèle unique est prédéfini, soit trop complexes, dans le cas où l'utilisateur définit lui-même son propre modèle de données et se force alors à maîtriser des langages de programmation. Le prototype Anthémis, quant à lui, propose de fournir un outil s'insérant entre ces deux extrêmes. Celui-ci doit pouvoir fournir à l'utilisateur un moyen de créer son propre modèle de connaissances sans pour autant maîtriser des langages de programmation complexes. Le prototype élaboré permet donc de formaliser les connaissances de l'utilisateur et de les stocker dans une base de données. Il offre la possibilité à l'utilisateur d'insérer son propre modèle de connaissances et ensuite d'intégrer ses propres connaissances dans le modèle défini par ses soins. Cette insertion se réalise, comme nous le verrons dans le sixième et le septième chapitre de ce présent travail, au moyen de fichiers XML.

Le second objectif se formule dans la possibilité de gérer les connaissances une fois celles-ci insérées dans la base de données. Par gestion des connaissances, nous entendons toutes les manipulations de création de nouvelles connaissances, de suppression des connaissances obsolètes ou tout simplement de modification des connaissances.

Le troisième objectif du prototype, quant à lui, est de permettre la diffusion des différentes connaissances se trouvant dans la base de données aux personnes qui désirent y accéder. Nous ne saurons présenter cet objectif car nous n'avons pas travaillé sur cette partie lors de notre stage.

Nous allons maintenant reprendre les trois buts de la gestion des connaissances présentés dans le TABLEAU 3.1. Nous observerons ainsi que ces trois buts correspondent aux trois objectifs présentés ci-dessus et nous allons illustrer les différents écrans de notre prototype en rapport avec ces trois buts.

Collecte des connaissances	La collecte des connaissances est effectuée. En effet, les connaissances sont insérées dans une base de données. Nous verrons, par la suite, le principe d'insertion mis en œuvre par notre prototype.
Production de connaissances	Ce but est concrétisé au sein de l'interface du prototype qui permet aux divers employés de créer de nouvelles connaissances au sein d'un modèle existant.
Diffusion des connaissances	Ce but est lui aussi atteint étant donné que le prototype est prévu pour tourner sur Internet pour permettre aux employés de visualiser les connaissances des autres.

TABLEAU 3.1 : Les buts de la gestion des connaissances repris par le prototype.

2.1. Collecte des connaissances

La collecte des connaissances se réalise par le chargement dans le prototype de fichiers contenant ces connaissances. Le principe de chargement des fichiers ne diffère pas d'une approche à l'autre. Par contre, comme nous le verrons dans le sixième et le septième chapitre, la structure de ces fichiers diffère ainsi que la méthode d'insertion dans la base de données.

Ainsi, le principe général reste le même : l'utilisateur écrit dans un fichier XML ses connaissances et charge ce fichier dans le prototype. L'insertion et l'agencement dans la base de données se font alors de façon automatique.

Selon l'approche Entité-Association, comme nous le verrons dans le sixième chapitre, l'insertion se réalise en deux temps : tout d'abord, insertion du modèle de connaissance et ensuite, insertion des connaissances proprement dites. Dans l'approche selon le formalisme des graphes conceptuels, comme nous le détaillerons dans le septième chapitre, l'insertion se fait, quant à elle, en trois temps : tout d'abord, insertion de la hiérarchie des concepts ensuite, insertion de la hiérarchie des relations et finalement insertion des connaissances.

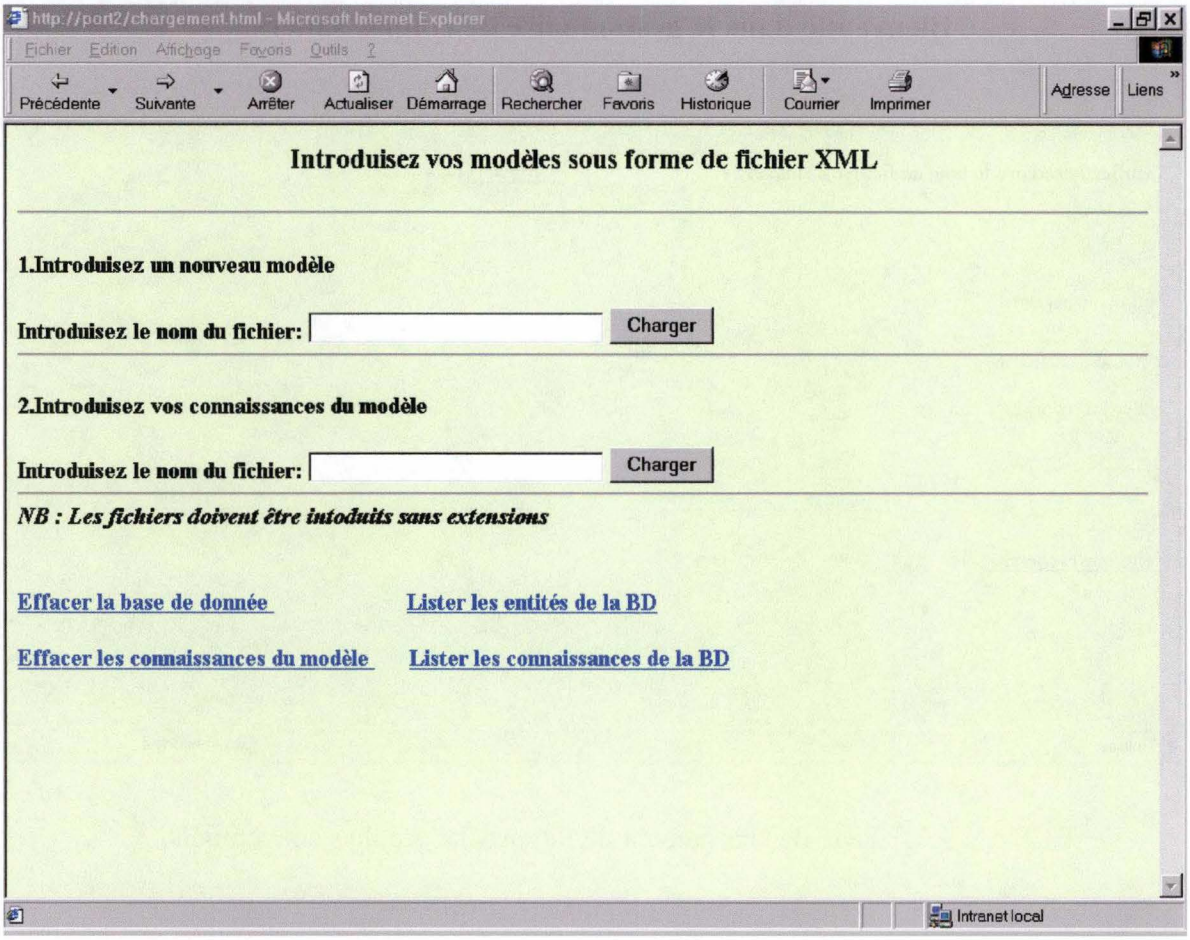
Cette collecte des connaissances se réalise grâce à la page de chargement qui permet aux utilisateurs de charger dans le prototype les fichiers en correspondance avec l'approche utilisée. Illustrons maintenant cette page.

2.1.1. Page de chargement

Suivant l'approche Entité-Association, la page de chargement, comme nous pouvons le voir sur l'ECRAN 3.1, offre à l'utilisateur les fonctionnalités suivantes : premièrement, les deux fonctionnalités les plus importantes sont le chargement dans la base de données d'un fichier reprenant le modèle et d'un fichier reprenant les connaissances liées à ce modèle. Deuxièmement, l'utilisateur peut effacer la base de données ainsi que les connaissances du modèle, cela veut dire qu'il peut vider les différentes tables de la base de données. Bien entendu, ces deux fonctionnalités n'ont aucun sens pour l'utilisateur, elles disparaîtront donc et n'ont été créées que par souci d'aide à la programmation. Troisièmement, il peut lister les entités du modèle ainsi que les connaissances.

Il faut noter que cette page et celles qui vont suivre pour cette approche ont été réalisées sans tenir compte de la possibilité de travailler sur plusieurs modèles simultanément. Pourquoi ne pas avoir travaillé sur le prototype en considérant la possibilité d'avoir plusieurs modèles dans la base de données ? Simplement car notre rôle dans le

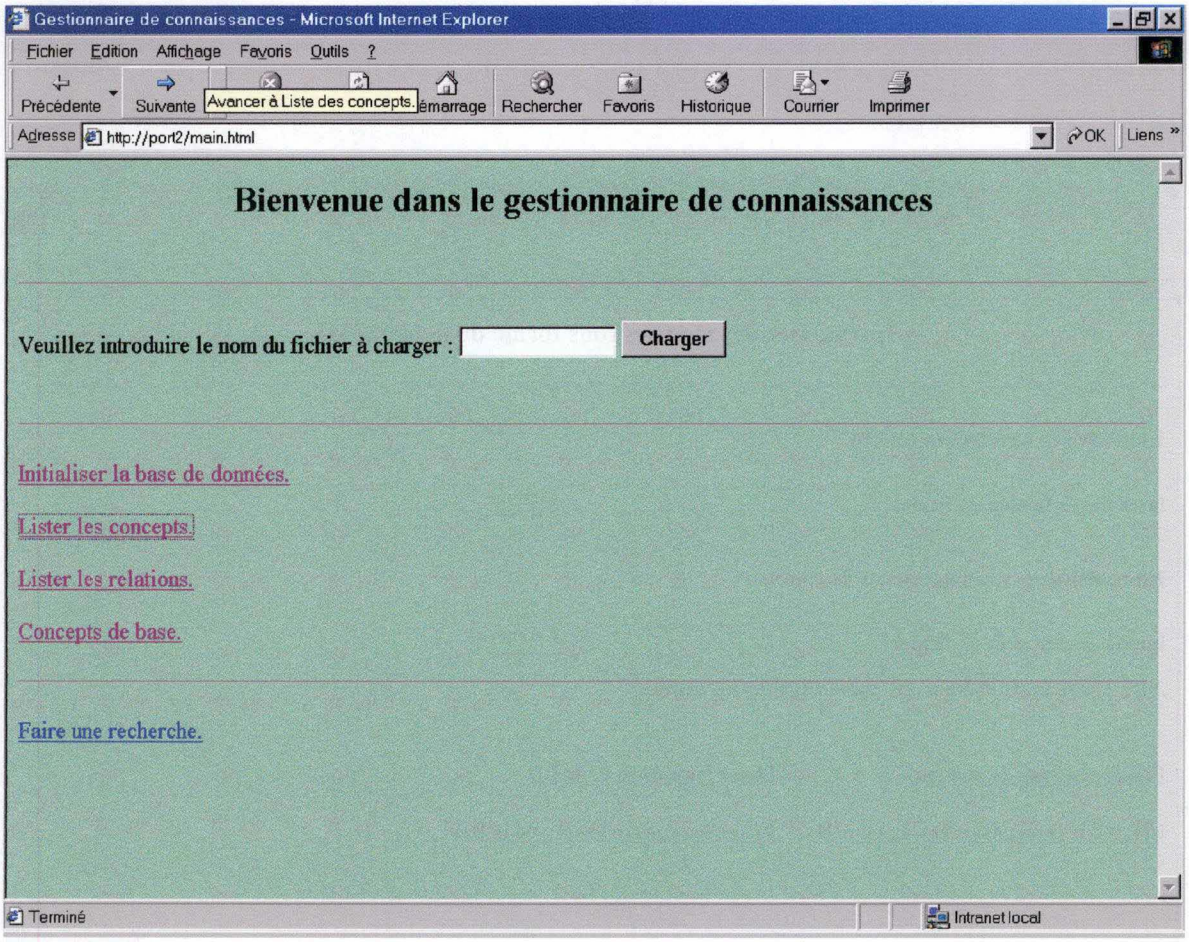
prototype était, avant tout, de vérifier que les objectifs du Professeur Gerbé étaient réalisables. Nous avons donc travaillé sur un seul modèle, tout en sachant que l'insertion de plusieurs modèles ne poserait aucun problème vu la structure de la base de données. En effet, si nous examinons le fichier initBD.asp se trouvant dans l'ANNEXE 2 de ce présent travail, nous pouvons remarquer que l'identifiant de la table TX_MODEL se retrouve dans toutes les autres tables. Nous pourrons donc dans l'avenir travailler avec plusieurs modèles simultanément sans problème d'intégrité. Nous reparlerons dans le huitième chapitre de ce travail d'une évolution graphique permettant de travailler aisément avec plusieurs modèles dans la base de données.



ECRAN 3.1 : Page de chargement de l'approche Entité-Association.

Suivant l'approche graphes conceptuels, la page de chargement du prototype diffère de celle précédemment présentée. En effet, l'utilisateur peut opérer plusieurs choix. (ECRAN 3.2). Tout d'abord, il peut charger un fichier dans la base de données. Ce fichier peut être soit une hiérarchie de concepts ou de relations, soit un fichier de connaissances.

Ensuite, il peut, comme pour l'approche Entité-Association, effacer la base de données. Il peut lister les concepts de base contenus dans la base de données, concepts de base qui permettront par après d'insérer des nouveaux fichiers de connaissances. Il peut également lister les concepts et relations présents dans son modèle. Enfin, il peut effectuer une recherche dans la base de données. Soulignons simplement que ces notions de concepts de base et de recherche dans le prototype seront explicitées dans le septième chapitre.



ECRAN 3.2. : Page de chargement de l'approche graphes conceptuels.

Il est évident que certaines fonctionnalités, comme celle permettant d'effacer la base de données, disparaîtront à long terme. Celles-ci ne sont présentes que pour tester le fonctionnement du prototype.

2.2. Production de connaissances

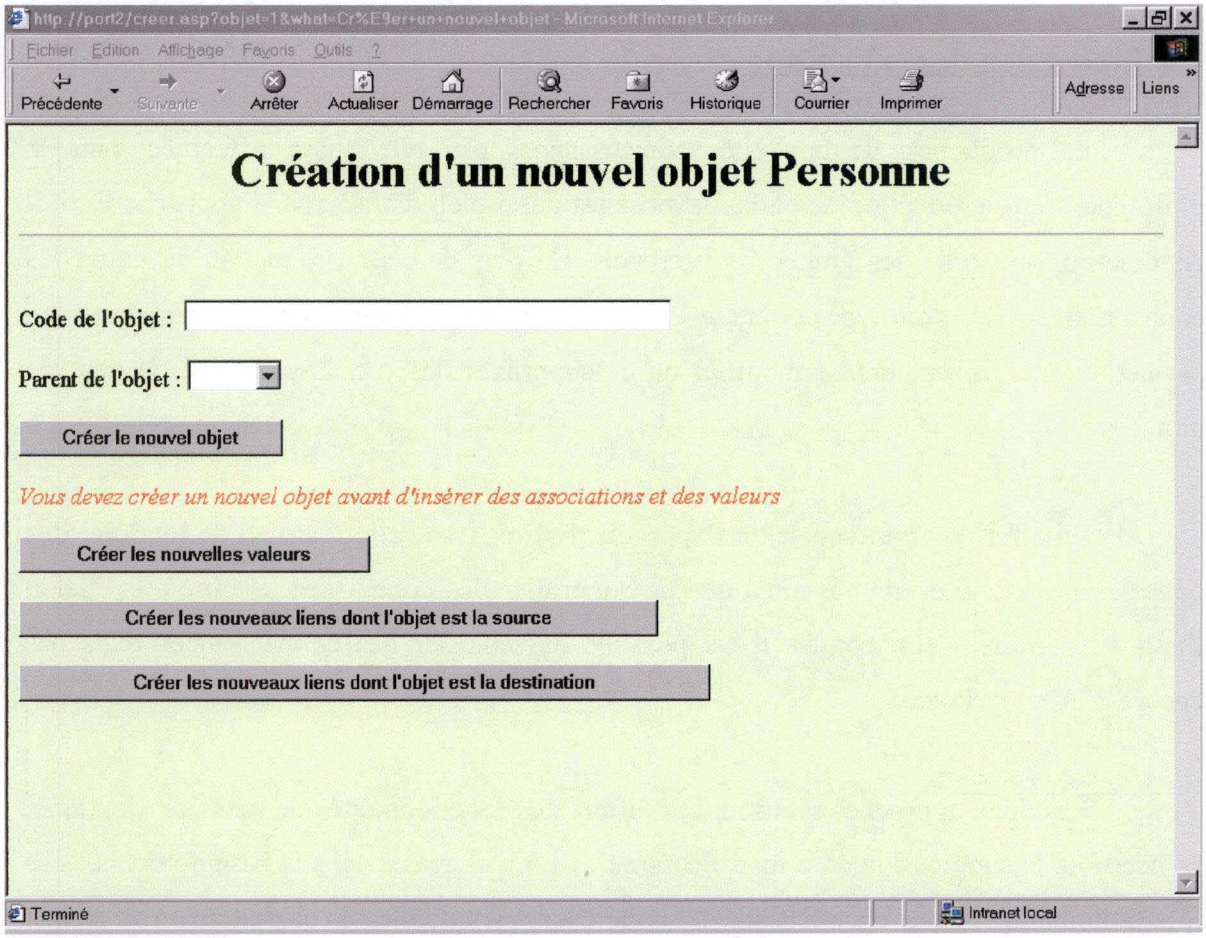
La production de nouvelles connaissances est, elle aussi, effectuée dans le prototype. Cette production de connaissances est aussi bien réalisée par l'approche Entité-Association que celle des graphes conceptuels. En plus de cette production de nouvelles connaissances, le prototype poursuit son objectif de gestion des connaissances en permettant aux utilisateurs de modifier ou de supprimer des connaissances de la base de données.

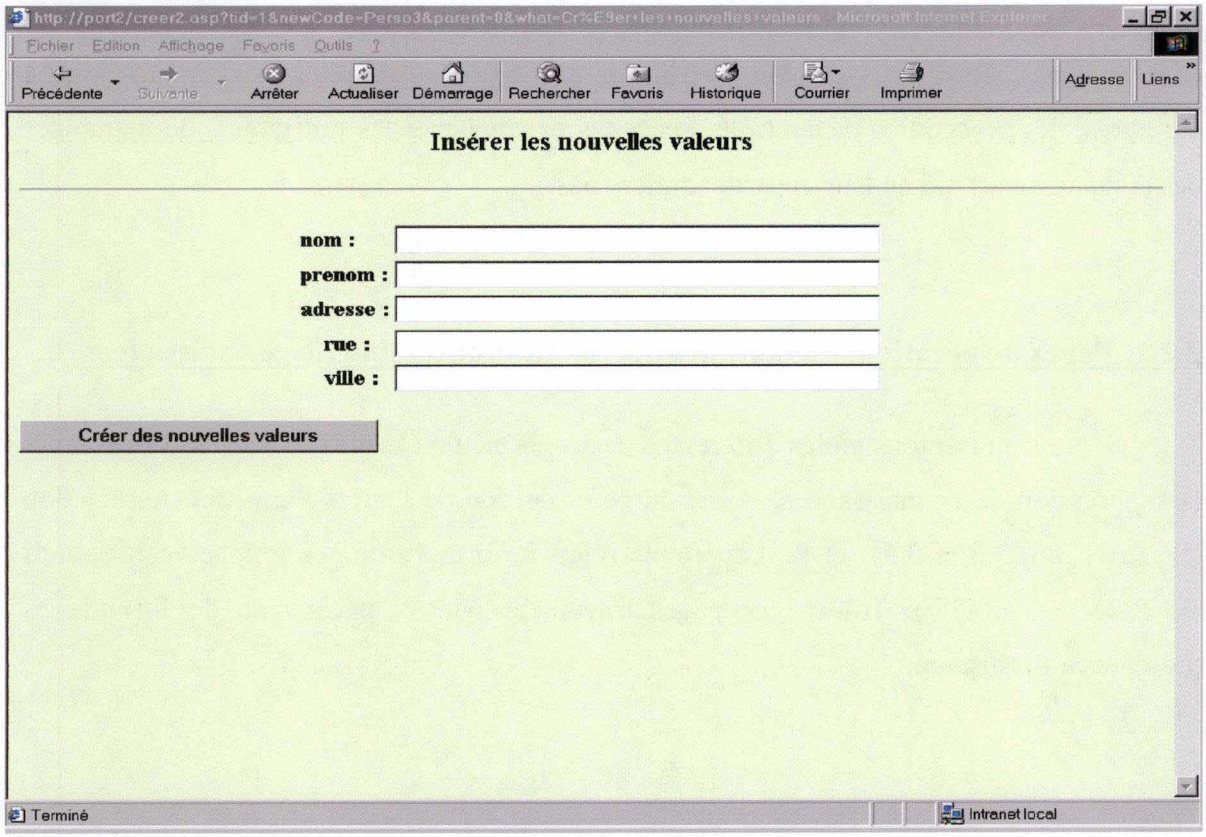
En effet, le prototype selon l'approche Entité-Association permet de modifier des connaissances déjà existantes ainsi que de supprimer des connaissances obsolètes. Selon l'approche graphes conceptuels il est possible de modifier des connaissances mais pas encore d'en supprimer.

Les deux approches réalisent ces différentes fonctionnalités de manière identique. L'insertion, la suppression et la modification des connaissances dans la base de données se font automatiquement en vérifiant la cohérence syntaxique. L'utilisateur n'a pas la possibilité de réaliser des changements sur ses modèles, il peut seulement agir sur les connaissances liées à ceux-ci. Nous verrons dans le huitième chapitre qu'il faudra ajouter, par après, des procédures de contrôle d'accès pour empêcher les utilisateurs de manipuler des connaissances qui ne leur appartiendraient pas.

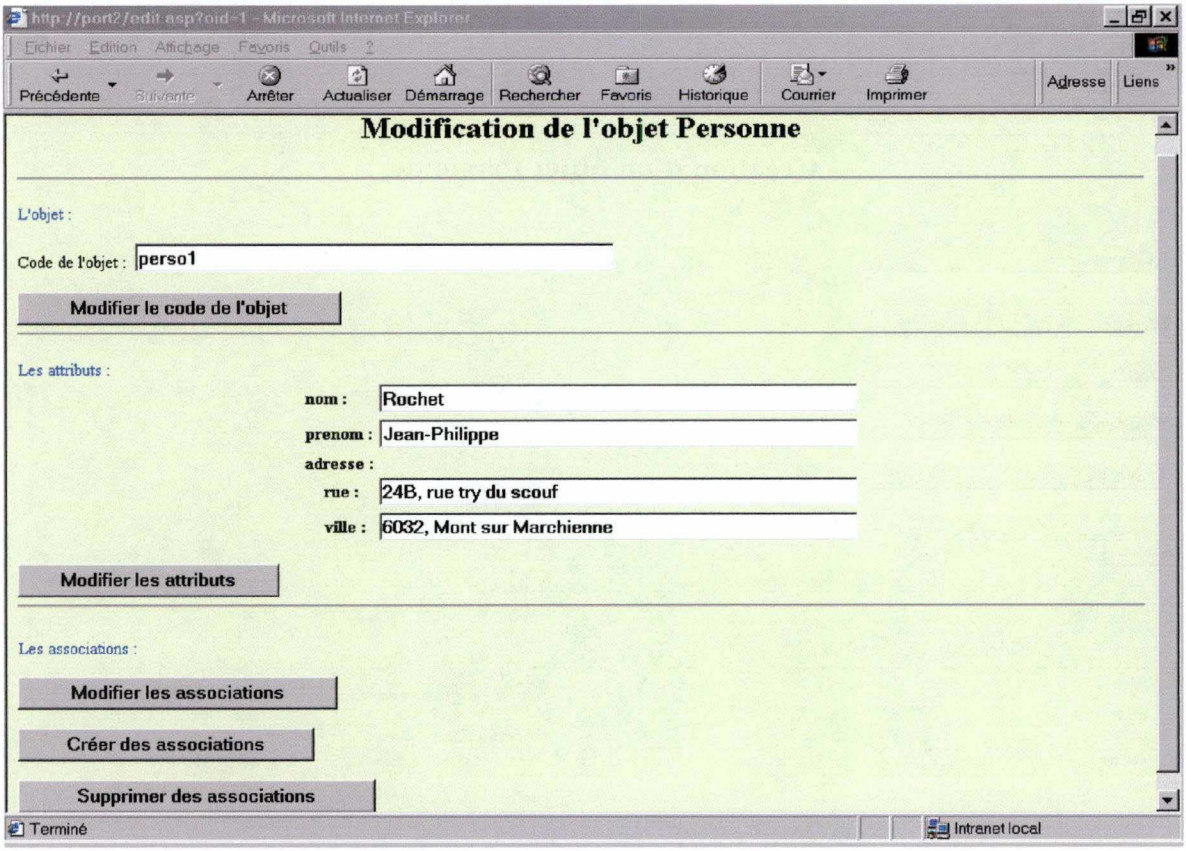
2.2.1. Pages de création, de modification et de suppression de connaissances

Nous présentons ici les différentes pages permettant la création, la modification et la suppression de connaissances, et ce, du point de vue de l'approche Entité-Association (ECRAN 3.3, 3.4 et 3.5) et de l'approche selon le formalisme des graphes conceptuels (ECRAN 3.6 et 3.7). Toutes ces pages travaillent par le mécanisme de formulaires présentés à l'utilisateur.

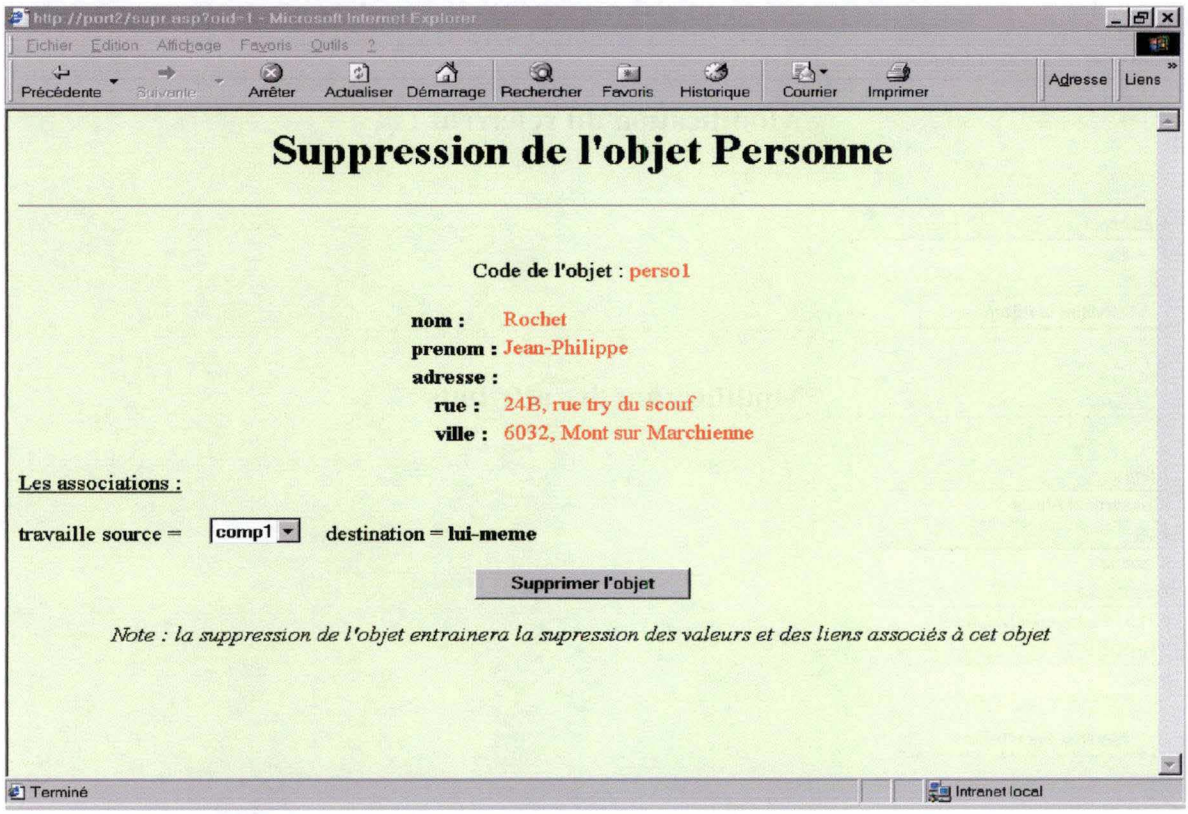




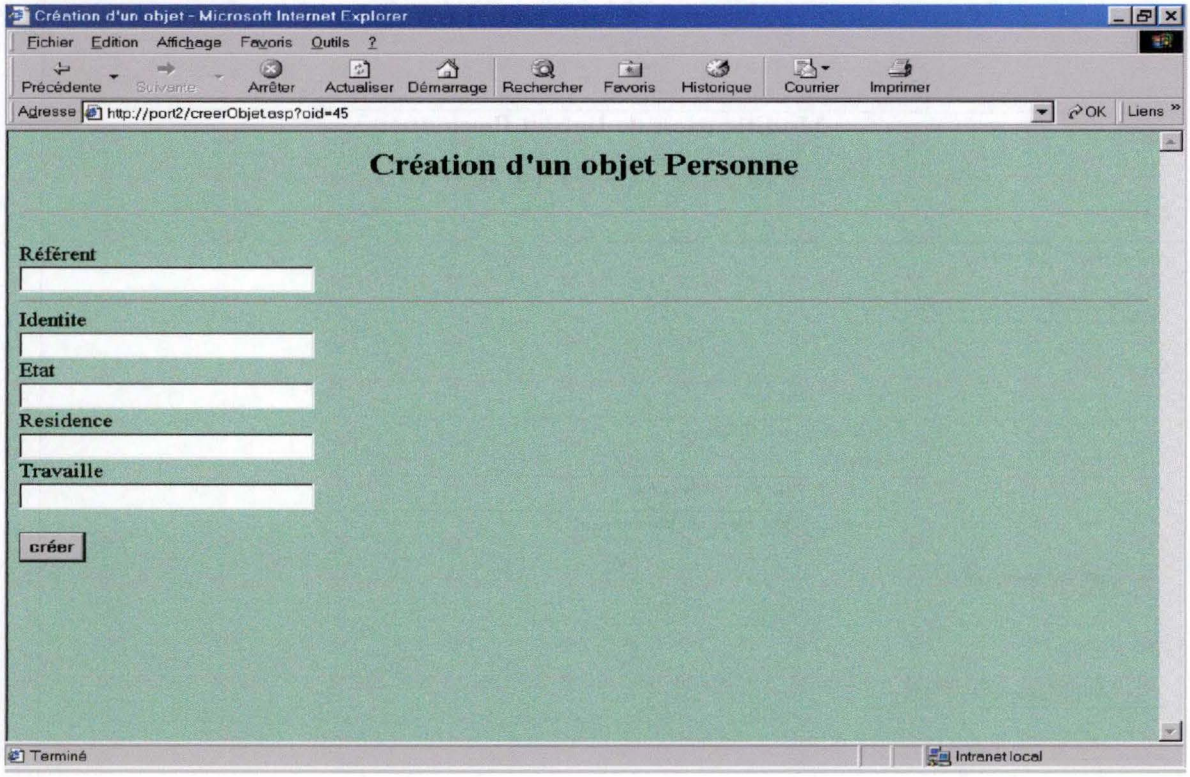
ECRAN 3.3 : Création d'un nouvel objet et insertion de ses valeurs.



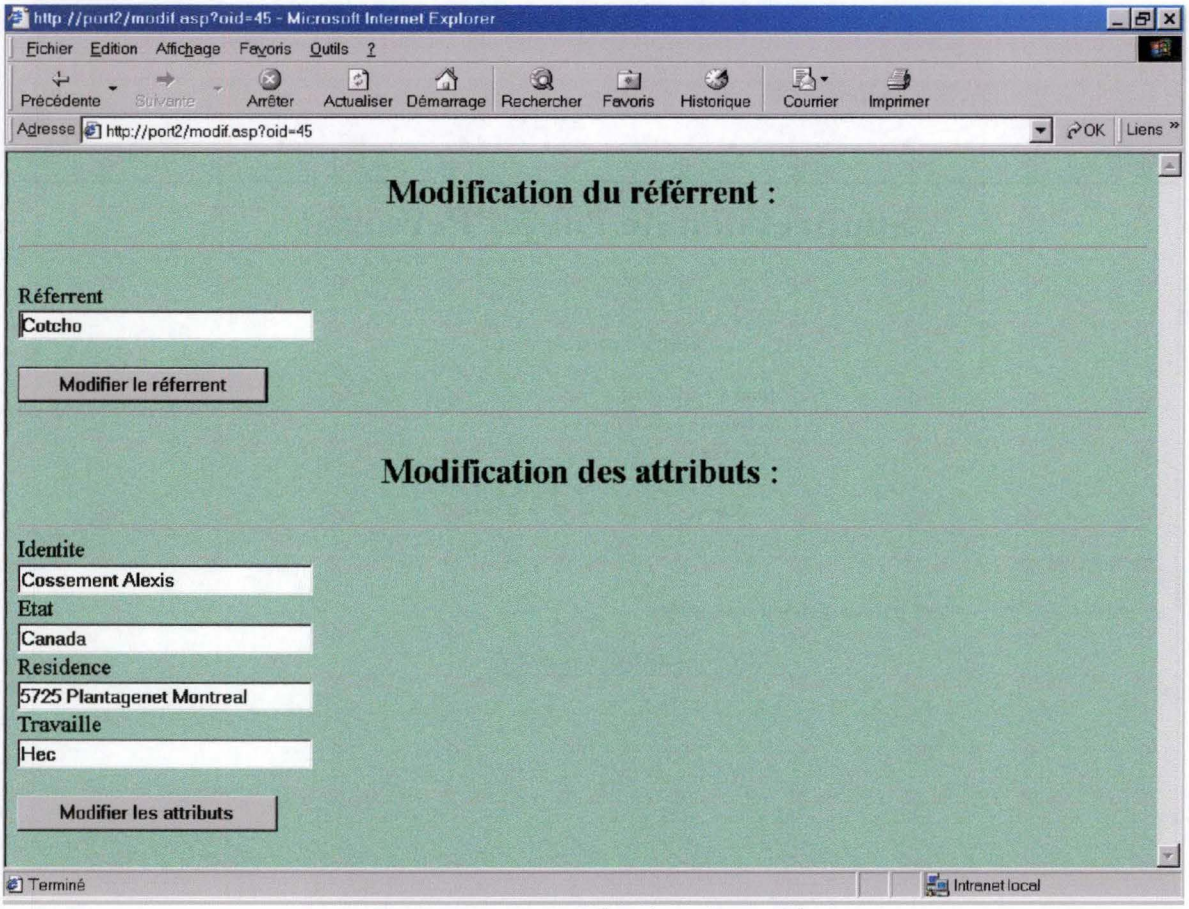
ECRAN 3.4 : Modification d'un objet et de ses associations.



ECRAN 3.5 : Suppression d'un objet.



ECRAN 3.6 : Création d'un nouvel objet.



ECRAN 3.7 : Modification d'un objet.

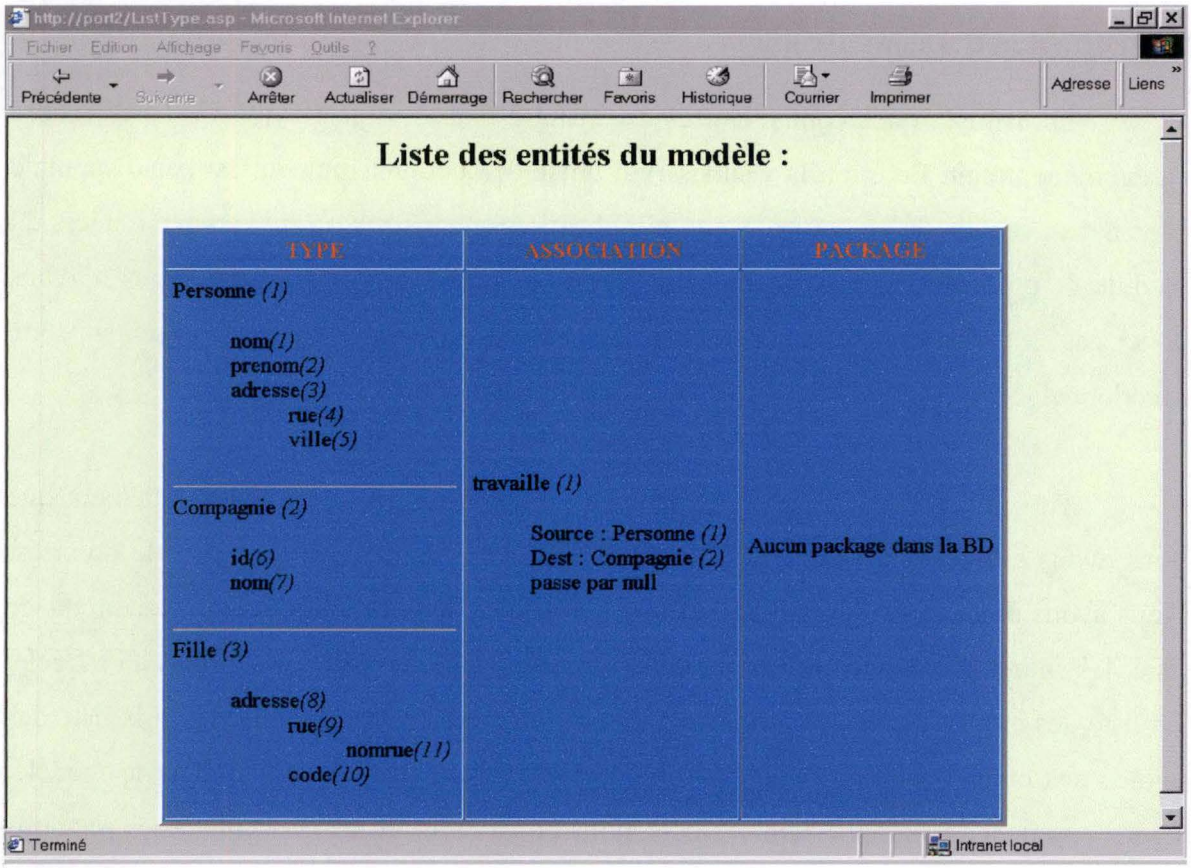
2.3. Diffusion des connaissances

La diffusion des connaissances est quant à elle un objectif qui n'est pas encore entièrement atteint. En effet, la visualisation finale de la connaissance n'est pas disponible dans notre version du prototype. Cette visualisation se construit au moyen des fichiers du module de présentation et ce module ne faisait pas partie de notre travail. Nous n'avons donc pas les fichiers de ce module et nous ne savons donc pas présenter cette fonctionnalité.

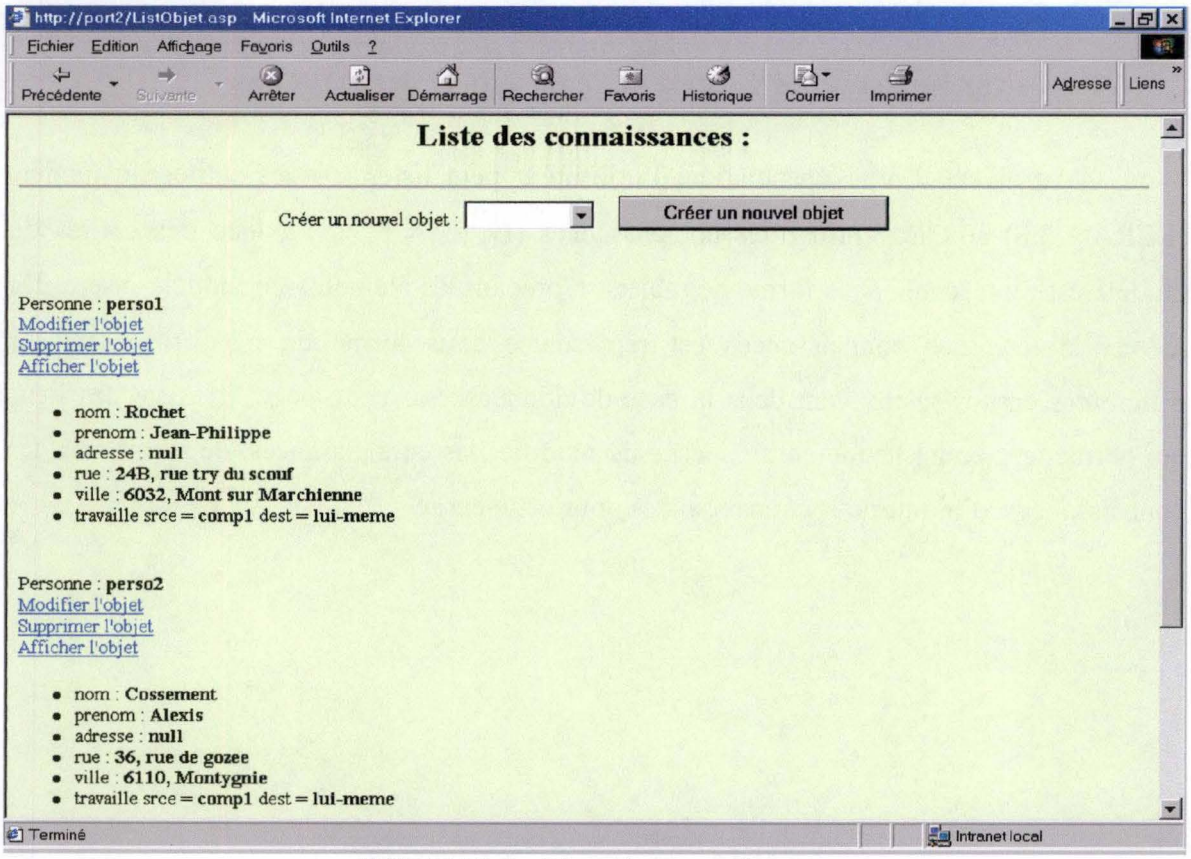
Par contre nous pouvons présenter pour cet objectif les différents fichiers que nous avons créés qui permettent de lister les différents éléments de la base de données. Nous avons donc, pour l'approche Entité-Association, deux fonctionnalités permettant de lister les entités du modèle de l'utilisateur et les entités des connaissances liées au modèle. La liste des entités du modèle donne à l'utilisateur une idée du modèle écrit et la liste des entités des connaissances donne l'ensemble des connaissances inscrites dans le modèle. Pour l'approche des graphes conceptuels, nous pouvons lister les concepts et les relations du modèle.

2.3.1. Pages sur les listes des entités, des concepts et des relations

Au niveau Entité-Association, l'utilisateur peut lister soit les entités du modèle (ECRAN 3.8) soit les entités des connaissances (ECRAN 3.9). La liste des entités du modèle est représentée sous forme de tableau reprenant les éléments du modèle inséré. De même, la liste des connaissances est représentée sous forme de page reprenant les différentes entités se trouvant dans la base de données. Sur cette page, il existe des liens qui permettent pour chaque entité décrite de modifier les connaissances, de supprimer des connaissances, d'ajouter des connaissances pour cette entité.

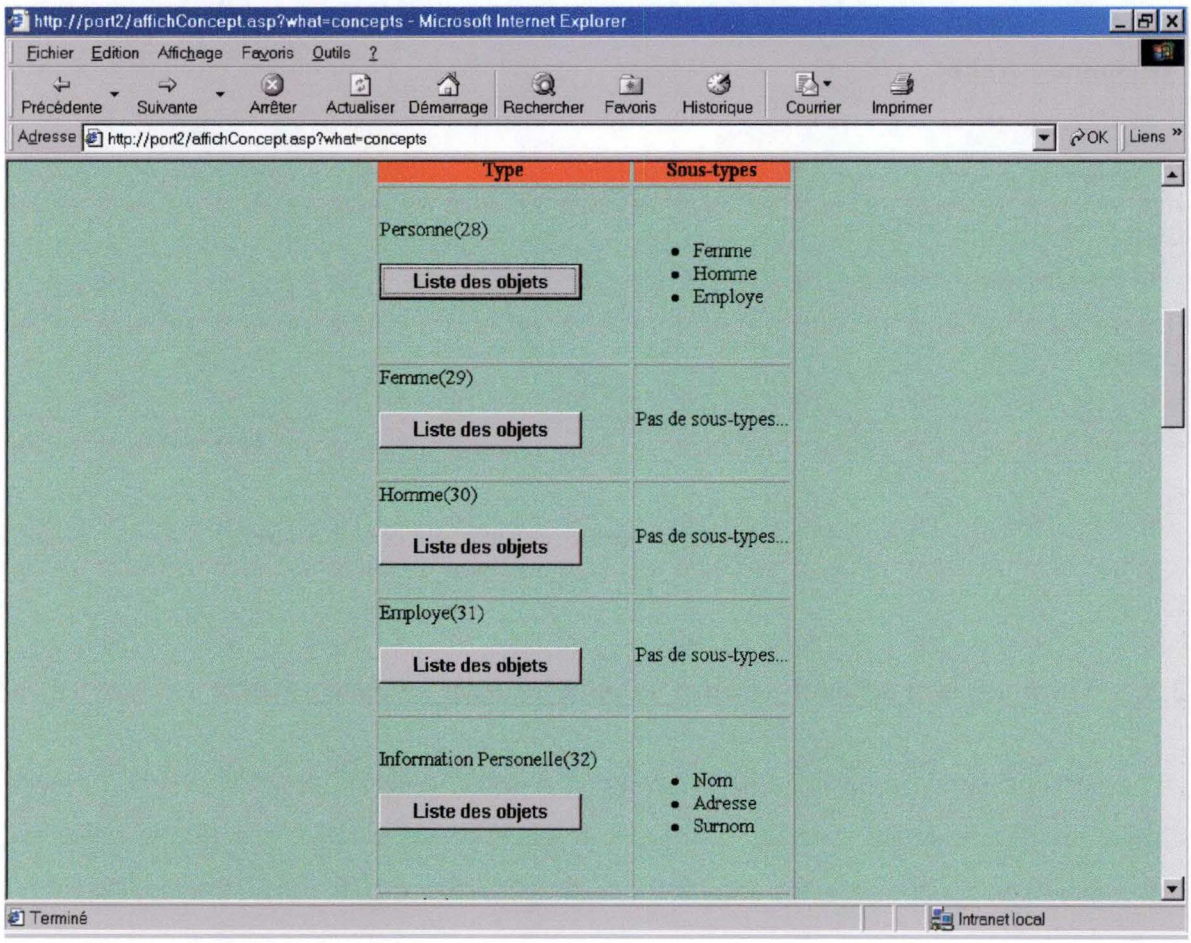


ECRAN 3.8 : Liste des entités du modèle.

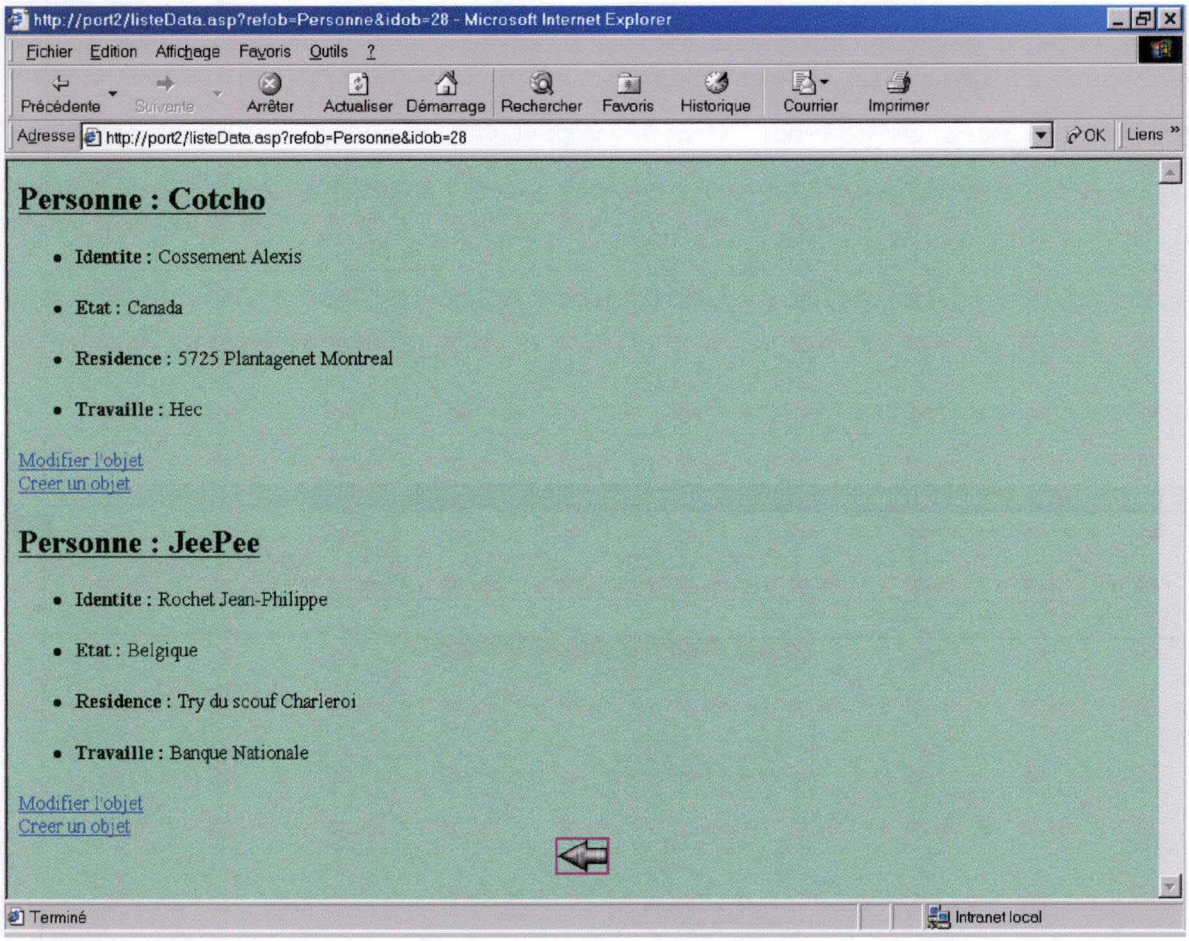


ECRAN 3.9 : Liste des connaissances.

Au niveau graphes conceptuels, la liste des concepts de la base de données se retrouve sous forme d'un tableau reprenant ces concepts (ECRAN 3.10). Ceux-ci sont organisés en hiérarchie de types. Nous avons pour chaque concept la liste de ses sous-types. A la différence de l'approche précédente, le lien permettant d'accéder à la page listant les différentes instances d'un concept particulier est un bouton. De plus, les opérations d'ajout et de modification se retrouvent sur la nouvelle page ainsi ouverte. Cette page reprend donc toutes les instances d'un concept particulier (ECRAN 3.11). La liste des relations suit le même principe que la liste des concepts présentée ci-dessus.



ECRAN 3.10 : Liste des concepts présents dans la base.



ECRAN 3.11 : Liste des instances d'un concept particulier.

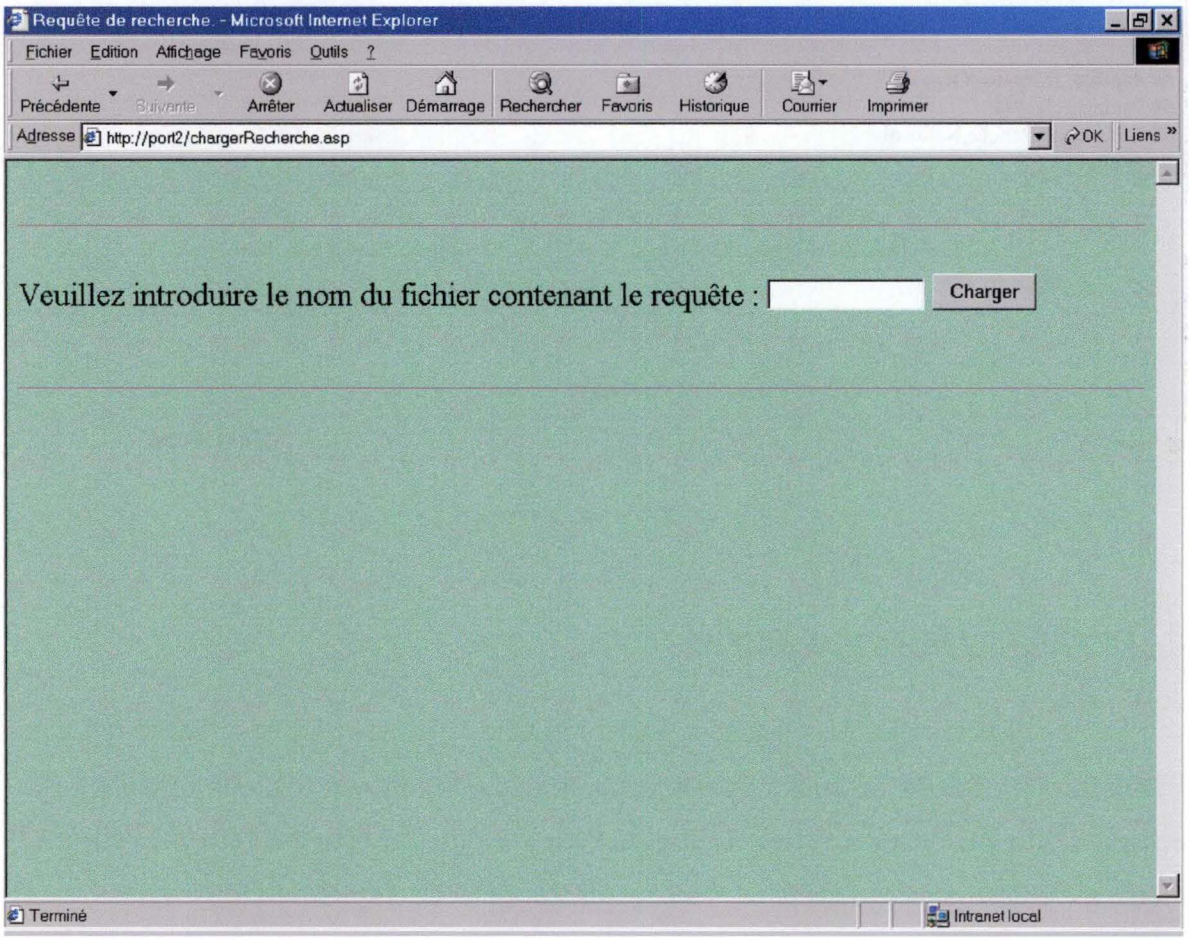
2.3.2. Méthode de recherche dans l'approche graphes conceptuels

Nous avons vu qu'une différence majeure entre les deux approches est la présence d'un outil de recherche dans le prototype graphes conceptuels. Nous allons présenter brièvement son fonctionnement dans cette section. Notons que le développement d'un tel outil pour l'approche Entité-Association ne poserait pas de problèmes mais sa puissance ne serait pas si importante.

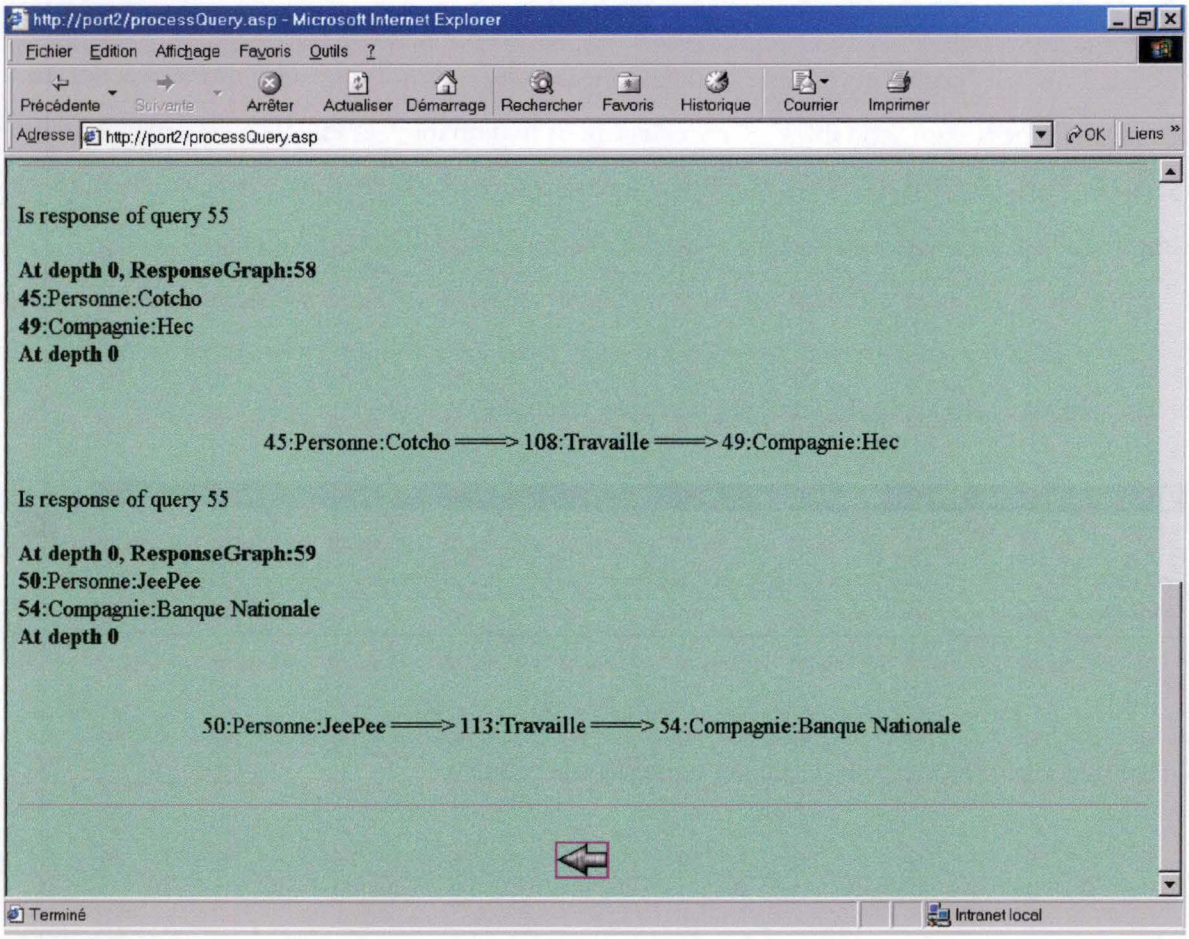
Lorsque l'on veut effectuer une recherche à partir de l'écran de chargement, il suffit de cliquer sur le lien adéquat. Une nouvelle page s'ouvre alors et l'utilisateur doit insérer son fichier de requête. (ECRAN 3.12). Le fichier est lui aussi écrit en XML et s'apparente à un fichier contenant de la connaissance. La différence est que certaines données sont laissées indéfinies. En effet, une particularité de ce formalisme est qu'il

accepte de la connaissance partielle, c'est-à-dire que le fichier introduit contient des valeurs nulles pour certaines connaissances. Le prototype va alors rechercher séquentiellement dans les tables les connaissances manquantes et les afficher à l'écran sous forme linéaire. (ECRAN 3.13). Sur cette page, nous voyons la réponse à une question toute simple qui est de savoir quels employés travaillent pour quelles compagnies.

Nous exposerons plus longuement dans le septième chapitre cet outil de recherche et nous verrons s'il répond bien à la puissance attendue.



ECRAN 3.12 : Chargement d'une requête.



ECRAN 3.13 : Résultats d'une requête.

3. Interface Homme-Logiciel

Le logiciel une fois terminé a pour objectif d'être sans pré-requis technique pour l'utilisateur. Il faut que le logiciel laisse la possibilité à l'utilisateur de s'occuper de tout et cela sans administrateur système. L'utilisateur devra donc insérer, gérer et visualiser lui-même ses connaissances sans pour autant avoir recours à des pré-requis.

Pour l'instant, n'étant qu'au stade de prototype, le logiciel requiert encore beaucoup d'expertise de la part de l'utilisateur. En effet, en ce qui concerne l'insertion de connaissance, l'utilisateur doit suivre trois étapes principales. Tout d'abord, l'utilisateur doit pouvoir représenter ses connaissances sous forme de modèle Entité-Association ou sous forme de graphes conceptuels. Ensuite, celui-ci doit écrire un fichier XML reprenant son modèle. Finalement, il doit écrire un second fichier XML reprenant ses connaissances à intégrer dans son modèle. Pourquoi utiliser le langage XML ? Simplement car le langage

XML, qui sera abordé dans le cinquième chapitre, est un langage adéquat pour l'échange de données.

Les pré-requis demandés à l'utilisateur sont donc la capacité de modéliser ses connaissances sous forme de modèle Entité-Association ou graphes conceptuels ainsi qu'une bonne connaissance du langage XML. Ces différentes exigences devront à terme être abolies pour répondre à l'objectif décrit ci-dessus.

Une des solutions envisagée par le Professeur Gerbé pour remplacer le pré-requis touchant au langage XML pourrait être l'utilisation d'un programme permettant de traduire des modèles Entité-Association ou des graphes conceptuels en fichier XML. En effet, il est possible d'écrire des logiciels permettant de passer d'une représentation de modèle Entité-Association en un fichier XML. C'est notamment le cas du programme DB-Main[2]. Au niveau des graphes conceptuels, notons qu'il existe une démarche automatique permettant de passer d'un graphe conceptuel à sa notation en XML et vice-versa. Celle-ci a été développée par un étudiant nous précédant sur le prototype, elle reste cependant fortement à un stade expérimental.

Il faut aussi savoir que le prototype vérifie la cohérence syntaxique des connaissances insérées dans la base de données et non la cohérence sémantique. En d'autres termes, cela signifie qu'une connaissance sera validée car elle se conforme au modèle prédéfini par l'utilisateur, mais elle ne sera pas rejetée si elle s'avère être une connaissance erronée, c'est-à-dire contradictoire par rapport à la réalité. Il est nécessaire de mettre en place un système matériel ou humain permettant de vérifier cette cohérence sémantique. Un système humain entraînerait la création d'un nouveau poste dans l'organisation qui serait un poste d'administrateur système. Nous reviendrons sur ce point dans le huitième chapitre de ce travail.

[2] DB-Main est un projet de recherche et de développement dans le domaine des applications des bases de données et de la technologie CASE. Ce projet est dirigé par Jean-Luc Hainaut de l'Institut d'Informatique à Namur.

4. Fiche d'identité du prototype

Nous allons maintenant présenter la fiche d'identité suivant la même structure que les fiches d'identité des diverses méthodes vues dans le deuxième chapitre du présent travail. Cette fiche d'identité renferme les éléments importants concernant le prototype.

Nom :	Anthémis.
Auteur(s) :	Olivier Gerbé.
Objectifs du logiciel :	Offrir un logiciel de capitalisation, gestion et consultation des connaissances sans pré-requis spécifiques pour l'utilisateur.
Première application du logiciel :	Ce logiciel est toujours actuellement à l'état de prototype.
Date d'élaboration :	2000
Pays d'origine :	Canada (Québec).
Description du prototype :	Constitution d'une base de connaissances de façon automatique et sans connaissance d'un quelconque langage de programmation, diffusion de la connaissance via Intranet et Internet. Uniformité de la présentation des connaissances.
Avantages :	Aucun pré-requis nécessaire à long terme pour l'utilisateur. Utilisation d'un formalisme puissant dans le prototype.
Faiblesses :	Pas encore développé, reste un prototype, on ne sait donc pas quelle sera sa valeur réelle après son implémentation et surtout lorsqu'il sera adapté en fonction des graphes conceptuels.
Intérêts :	Présentation des connaissances de manière standardisée et structurée. Navigation à travers celles-ci de manière aisée grâce à la génération automatique de liens hypertextes. Base de données flexibles permettant l'utilisation de modèles écrits par les utilisateurs.

FICHE 3.1 : Fiche d'identité du prototype Anthémis.

Conclusion de chapitre

Nous avons vu dans ce chapitre une description générale du prototype Anthémis de gestion des connaissances que nous avons développé durant notre stage. Le prototype Anthémis issu d'une idée originale du Professeur Gerbé des HEC de Montréal se

développe selon deux approches : le modèle Entité-Association et le formalisme des graphes conceptuels. Cette dernière approche est l'objectif ultime du logiciel de gestion des connaissances. En effet, le prototype a pour but à long terme de valider la théorie proposée par le Professeur Gerbé dans sa thèse intitulée *'Un modèle uniforme pour la modélisation et la métamodélisation d'une mémoire d'entreprise'*. [GERBE00-1]

Le prototype suit essentiellement trois objectifs généraux qui sont : une insertion libre de modèles et de connaissances liées à ces modèles dans la base de données ; une gestion des connaissances de la base de données permettant la création, la modification et la suppression des connaissances ainsi qu'une présentation uniforme des connaissances de la base de données. Le prototype a pour but d'être le plus flexible possible et de laisser la plus grande liberté à l'utilisateur tout en exigeant, à long terme, peu de pré-requis. Pour le moment en effet, l'insertion des données nécessite la maîtrise du langage XML et une certaine capacité à modéliser ses connaissances sous forme de modèle Entité-Association ou de graphes conceptuels. Le prototype se veut auto-régulateur, c'est-à-dire qu'il veut laisser la possibilité à l'utilisateur de modéliser lui-même ses modèles de connaissances et ses connaissances liées à ces modèles ainsi que de naviguer aisément entre les différentes connaissances introduites dans le système.

Nous avons aussi présenté les différents modules que comporte notre prototype Anthémis ainsi que les interactions existantes entre ces modules. Nous avons alors signalé que nous n'avons pas travaillé sur l'entièreté du prototype. En effet, nous n'avons pas travaillé sur le module présentation et nous sommes donc dans l'impossibilité de présenter ce module et les fonctionnalités qui y sont liées.

Nous avons vu que les différentes fonctionnalités du prototype remplissent bien ses objectifs ainsi que les buts de la gestion des connaissances et nous avons illustré ces différentes fonctionnalités par ses différents écrans.

Nous allons dans les deux chapitres suivant détailler les différentes technologies employées par le prototype ainsi que les différents langages de programmation utilisés pour créer ce prototype.

Quatrième chapitre

Technologies utilisées dans le prototype Anthémis

I. Présentation de l'infrastructure utilisée

II. Explication des différentes technologies

1. Server-side

1.1. Base de données ORACLE

1.2. Serveur WEB

1.2.1. Comparaison Apache et IIS

1.2.2. Personnal Web Server

1.3. Active Server Page

1.4. ODBC

2. Client-Side

Introduction de chapitre

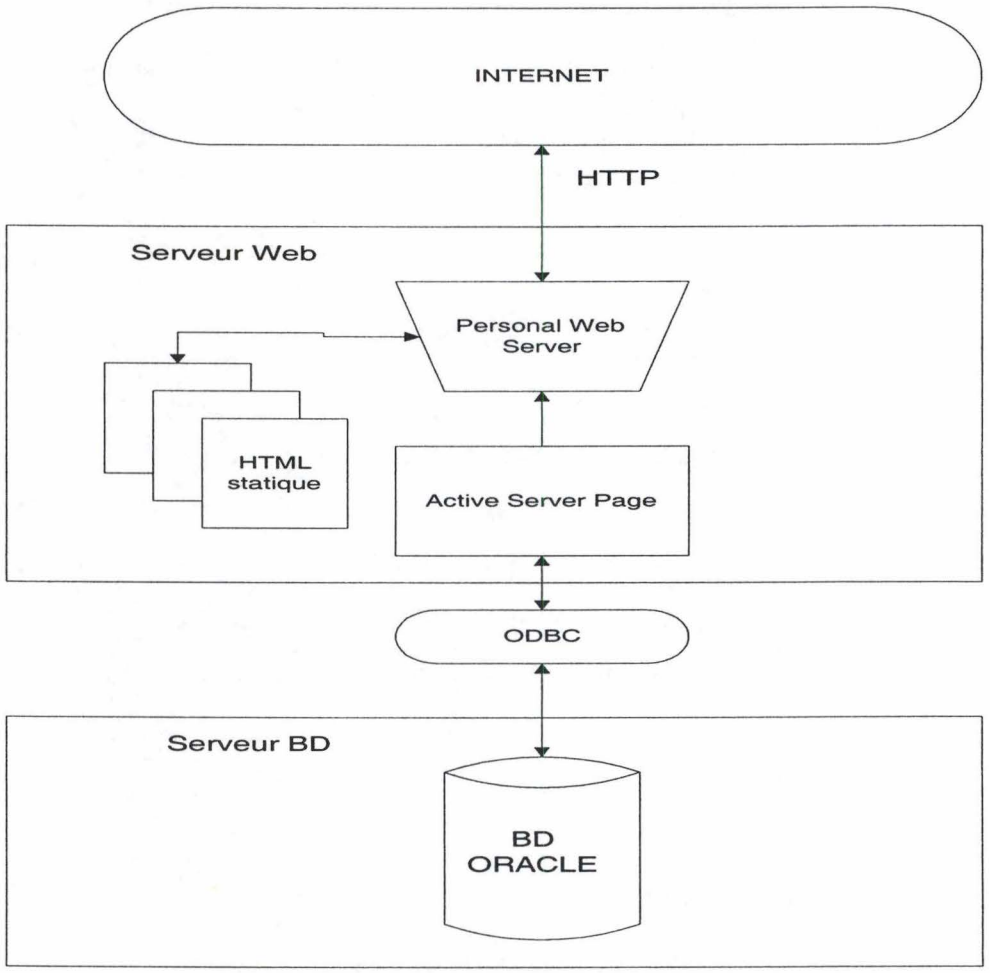
La création et l'utilisation d'un programme gérant les connaissances des entreprises nécessitent un ensemble de technologies. Le choix de ces technologies est important pour la qualité future du logiciel. Le but de ce chapitre est de décrire les différentes technologies utilisées pour le prototype Anthémis tant du côté client que du côté serveur.

Du côté serveur, une base de données relationnelle gérée par un système de gestion de bases de données a été utilisée. Le système choisi est ORACLE. En ce qui concerne le choix du serveur WEB, nous avons travaillé avec Personnal Web Server de Microsoft. Ce logiciel permet de simuler en local un serveur WEB. Nous examinerons les serveurs qui existent sur le marché et ferons une comparaison des plus courants, à savoir Internet Information Server de Microsoft et Apache du Apache Group. Du point de vue de la dynamique des pages HTML, nous avons utilisé la technologie Active Server Page (ASP) qui est, elle aussi, une technologie développée par Microsoft. Enfin, pour gérer la communication entre les applications clientes et le système de gestion de base de données, nous avons utilisé le protocole Open Data Base Connectivity (ODBC).

Du côté client, le prototype ne nécessite pour l'utilisateur que la simple possession d'un navigateur dont les plus répandus sont Internet Explorer et Netscape Navigator. Nous ne détaillerons pas le côté client vu la banalité des navigateurs. En effet, de nos jours tout le monde sait ce qu'est un navigateur.

I. Présentation de l'infrastructure utilisée

Il est important de rappeler que lorsque nous avons commencé à travailler sur le prototype Anthémis, deux autres étudiants avaient déjà contribué à sa conception sous la supervision du Professeur Gerbé. Celui-ci était en effet désireux de mettre au point un programme permettant de gérer les connaissances des entreprises. Lorsque nous nous sommes joints au projet, les bases avaient déjà été réalisées et le noyau central avait déjà été établi. La question concernant le choix des outils à utiliser avait donc déjà été prise en considération par ces intervenants. Quant à nous, nous avons dû continuer le travail de nos prédécesseurs en utilisant les technologies qui avaient été choisies au préalable. Pour comprendre l'agencement de ces différentes technologies, nous allons nous baser sur un schéma reprenant celles-ci. (SCHEMA 4.1)



SCHEMA 4.1 : Technologies utilisées pour Anthémis.

II. Explication des différentes technologies

Nous devions travailler sur un prototype capable de gérer d'un côté un serveur et de l'autre côté des clients pour permettre la diffusion des connaissances. Nous allons tout d'abord nous intéresser aux technologies se trouvant du côté du serveur et puis nous aborderons celles concernant le client.

1. Server-Side

Dans cette section, nous verrons les technologies qui se trouvent du côté du serveur et que nous avons utilisées pour créer le prototype Anthémis. Ces technologies sont : une base de données pour stocker les connaissances, un serveur WEB permettant la connexion à Internet et ASP (Active Server Page) permettant la gestion de pages WEB dynamiques. La compréhension de ces technologies est vitale pour acquérir une bonne vision du prototype Anthémis que nous décrirons par la suite.

1.1. Base de données ORACLE

Le prototype Anthémis utilise une base de données relationnelle centralisée qui se trouve pour l'instant sur le serveur des Hautes Etudes Commerciales de Montréal *HEC/TIM*. Cette base de données est gérée par un SGBD (Système de Gestion des Bases de Données) qui n'est autre qu'ORACLE. Toutes les manipulations sur les données sont réalisées suivant un langage unique qui est SQL (Structured Query Language).

Les bases de données nous donnent un certain nombre d'avantages indiscutables tels que :

- la mise à jour des informations,
- une bonne structure de ces informations,
- des outils gérant ces informations,
- la possibilité d'avoir une bonne vitesse d'accès à l'information.

Une base de données relationnelle stocke des informations dans des tableaux. Chaque tableau peut en général être relié aux informations d'autres tableaux afin d'apporter une réponse à une requête. Une base de données relationnelle est donc un ensemble de tables contenant des informations et dans lesquelles les lignes peuvent avoir des relations avec d'autres lignes appartenant à d'autres tables.

Une base de données ORACLE convient bien pour des organisations qui reçoivent beaucoup de requêtes. ORACLE est le promoteur de bases de données le plus important au monde. Il propose des bases de données pour Windows NT et différents systèmes d'exploitation UNIX comme LINUX et a créé son propre ensemble d'outils, ORACLE WEB Développer Suite. ORACLE est donc un système simple d'utilisation et est largement répandu. Le choix de ce dernier paraît dès lors très judicieux.

1.2. Le serveur WEB

Le serveur WEB est, comme son nom l'indique, un serveur qui permet d'établir des connexions avec Internet. Ce serveur doit être installé sur l'ordinateur qui contient la base de données. Il existe plusieurs serveurs WEB différents tels que Apache, IIS (Internet Information Server), Netscape Server ou bien encore Oracle WEB Application Server. Avant d'examiner plus précisément Apache et IIS, notons simplement que les deux derniers sont multi plates-formes et que le serveur Oracle est fortement lié à sa base de données. Nous avons choisi de décrire ces deux serveurs car ce sont eux qui sont les plus répandus comme le montre une étude que nous verrons par la suite. En ce qui nous concerne, nous avons utilisé Personnal Web Server (PWS) qui est dérivé d'Internet Information Server.

Les serveurs WEB consistent à analyser les requêtes des clients, à chercher les informations demandées sur le serveur, à interagir avec la base de données et à faire parvenir ces informations aux clients via HTTP (Hyper Text Transfer Protocol). HTTP permet, de manière générale, le transport de fichiers entre un serveur WEB et une station cliente.

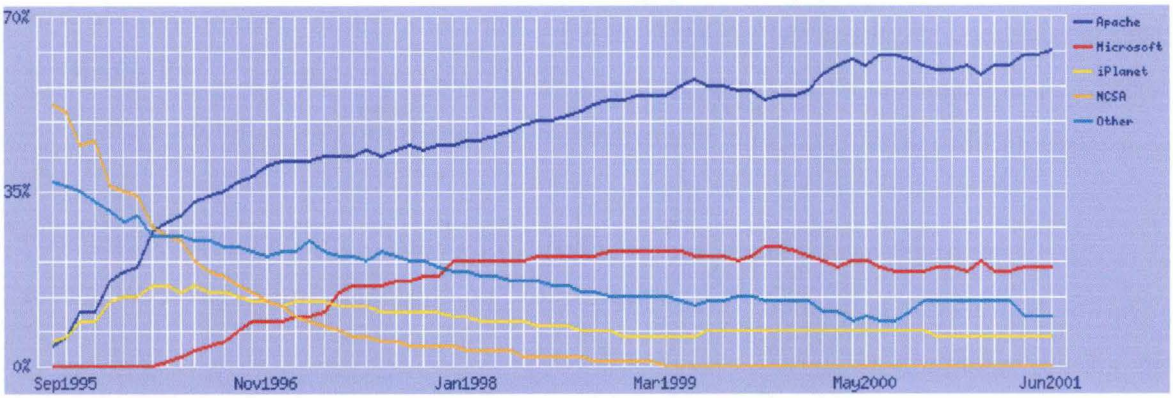
Ces serveurs WEB permettent d'accéder à des pages statiques telles que les pages écrites en HTML (Hyper Text Markup Language). Ces pages sont appelées des pages statiques tout simplement parce que ces pages sont écrites une fois pour toutes et sont rarement modifiées. C'est le cas des pages WEB qui ne sont pas connectées à une base de données. De plus, ces modifications doivent être faites à l'intérieur même du code HTML. Mais ces serveurs peuvent également donner accès à des pages actives telles que des pages de type ASP. Nous aborderons plus tard ce type de pages. Notons simplement que ces pages sont dites actives car elles permettent une interaction avec l'utilisateur et se forment suivant les actions de ce dernier.

1.2.1. Comparaison Apache et IIS

Au début de l'année 1995, le logiciel le plus populaire était le *daemon* HTTP développé par McCool. Ce logiciel a été élaboré au National Center for Supercomputing Applications (NCSA) de l'université de l'Illinois. Son développement s'est arrêté après le départ de McCool du NCSA et de nombreux webmasters ont dû eux-mêmes prendre en charge le développement du logiciel en y ajoutant leurs propres extensions et corrections de bugs. Une petite communauté s'est ainsi créée sous le nom d'Apache Group. Les objectifs de ce groupe sont de centraliser tous les développements apportés aux logiciels pour en faire bénéficier tous les utilisateurs. C'est ainsi que le logiciel Apache a vu le jour.

Internet Information Server est quant à lui un produit Microsoft qui a été conçu en 1995. Celui-ci regroupe IIS 3.0 tournant sous Windows NT, IIS 4.0 tournant sous Windows 2000 et Microsoft Personal Web Server pour Windows 95, 98 et 2000.

Ces deux logiciels étant apparus à peu près à la même époque, on peut faire une comparaison de leur évolution respective depuis septembre 1995 jusqu'en juin 2001. (SCHEMA 4.2)



SCHEMA 4.2 : Evolution des différents serveurs WEB. [NETCRAFT01]

On remarque sur ce schéma la nette domination du *daemon* HTTP de McCool, dénommée NCSA sur le schéma, avant l'arrivée d'Apache. Mais Apache, aidé dans son développement par les utilisateurs du monde entier, s'est rapidement taillé une importante part de marché et ce, contrairement à l'utilisation des produits Microsoft ou Netscape qui ne connaissent qu'une faible progression. A l'heure actuelle, selon une récente étude menée par Netcraft en juin 2001, on constate que la majorité des serveurs WEB sont Apache. (TABLEAU 4.1)

Server	June 2001	Percent
Apache	18466153	63.02
Microsoft-IIS	5972321	20.38
Netscape-Enterprise	1768673	6.04
Zeus	810108	2.76

TABLEAU 4.1 : Pourcentage de pénétration des différents serveurs.[NETCRAFT01]

La particularité principale d'Apache est d'être totalement gratuit. En effet, aucun centre de développement n'existe pour ce logiciel. Les améliorations sont apportées par les utilisateurs eux-mêmes. Chaque proposition d'évolution est examinée par les personnes ayant contribué au logiciel et est acceptée ou refusée. Cette caractéristique implique aussi le fait que les codes sources soient entièrement disponibles gratuitement.

Cette disponibilité des codes sources constitue un atout supplémentaire car elle permet aux utilisateurs d'adapter le logiciel directement à leurs besoins et d'optimiser son

utilisation. De plus, les utilisateurs peuvent garantir une compatibilité maximale avec leur matériel. Une autre particularité d'Apache est de fonctionner par modules. L'utilisateur est donc totalement libre de ses choix en concordance avec l'utilisation qu'il fait du logiciel.

IIS n'est, quant à lui, pas un logiciel libre et de ce fait, l'utilisateur paye pour son acquisition. Notons que la dernière version de IIS est gracieusement fournie avec Windows 2000 qui, lui, reste payant.

Une autre différence de taille est le système d'exploitation sous lequel le serveur WEB tourne. Si différentes versions du logiciel Apache sont disponibles pour plusieurs plates-formes comme Linux, Windows ou UNIX ; IIS n'est, quant à lui, disponible que pour Windows. De même, Apache supporte une grande variété de plates-formes matérielles tandis que IIS ne supporte que celles qui le sont par Windows.
(TABLEAU 4.2)

	Apache	**Microsoft-IIS**
Systèmes d'exploitation	NetBSD Digital UNIX BSDI AIX OS/2 SCO HPUX Windows NT Linux FreeBSD IRIX Solaris	Windows NT

TABLEAU 4.2 : Répartition des Systèmes d'exploitation. [SECK98, p11]

A part ces différences majeures, ces deux serveurs offrent à peu de choses près les mêmes fonctionnalités pour la gestion des pages HTML, la sécurité et l'administration. Nous avons vu que le développement s'est effectué avec Personnal Web Server mais le choix du serveur final reste encore une incertitude. Notons que si la technologie ASP reste utilisée par la suite, l'emploi de Internet Information Server est obligatoire car lui seul permet de gérer les pages de type ASP.

En conclusion, Apache apparaît comme plus intéressant car celui-ci est multi plates-formes et fonctionne dans des environnements hétérogènes. De plus, son implémentation en modules permet un libre choix dans son utilisation. Notons que, selon la politique établie, il est peut-être préférable de choisir un serveur payant comme IIS. En effet, le support technique sera beaucoup plus important et performant.

1.2.2. Personnal Web Server

Personnal Web Server (PWS) fourni par Microsoft offre les mêmes fonctionnalités qu'un serveur WEB, du moins au point de vue de la gestion des pages HTML et de l'accès à la base de données. Toutefois, la particularité de ce serveur est que tout se déroule au niveau local. En effet, PWS simule un serveur WEB et permet ainsi de tester ses pages WEB sans connexion. Tous les accès à la base de données se font donc localement.

De plus, tout le code des pages contenant des scripts peut lui aussi être testé grâce à la simulation du serveur alors que cela serait impossible sans PWS.

Le choix s'est donc porté naturellement sur PWS étant donné que la technologie ASP a été utilisée. PWS permet de tester la validité des pages ASP et des scripts inclus dans le code de ces pages. De plus, PWS est très simple d'utilisation. Il offre la simulation d'un serveur WEB avec un minimum de configuration, et de ce fait, un minimum de manipulation par l'utilisateur.

1.3. Active Server Page

Active Server Page (ASP) est un standard de Microsoft permettant de gérer dynamiquement des pages HTML. En effet, le mécanisme ASP comprend plusieurs phases. Le serveur reçoit tout d'abord du client une requête sous forme de fichier avec l'extension *.asp. Le serveur appelle donc le fichier ASP concerné et reçoit en retour une page HTML dynamique. Cette page est renvoyée au client en guise de réponse à sa requête. Les fichiers ASP peuvent contenir des langages de scripts comme Jscript ou

VBScript. Les différents scripts contenus dans ces fichiers sont alors exécutés sur le serveur.

De cette manière, on peut créer des scripts sur le serveur à partir de n'importe quel langage de scripts qui est compatible avec COM (Component Object Module) de Microsoft. En effet, ASP est une technologie propriété de Microsoft. De ce fait, cette technologie ne tourne que sur des serveurs Microsoft, d'où l'utilisation de Personnal Web Server.

Comme nous l'avons vu plus haut, un fichier ASP peut contenir des scripts. Mais il est fréquent de rencontrer aussi du texte et, dans la majorité des cas, du code et des balises HTML. Au point de vue pratique, les scripts se différencient du code HTML par des délimiteurs. Ceux-ci sont simplement les suivants : le début est indiqué par <% et la fin par %>. Voyons comment le serveur exécutera le code du SCHEMA 4.3.

```
<HTML>
    <BODY>
        Cette page a été actualisée pour la dernière fois <%=Now()%>.
    </BODY>
</HTML>
```

SCHEMA 4.3 : Exemple de code pour un fichier ASP. [GOYETTE99]

Lorsque le serveur exécute ce code, il remplace la fonction VBScript par l'heure et la date courante et renvoie le tout au navigateur. Il existe dans les fichiers ASP des directives spéciales. La première est la directive de traitement englobée dans les balises <%@ et %>. Celle-ci envoie au serveur les indications nécessaires pour traiter le fichier. Ainsi, on peut retrouver en début de fichier le nom du langage de script utilisé : *<%@ LANGAGE=JSCRIPT %>*. Les directives de sortie permettent quant à elles d'afficher la valeur d'une expression et se présentent comme suit : *<%= expression %>*.

Il faut aussi noter qu'ASP contient un ensemble d'objets intégrés qui facilitent l'envoi des requêtes et des réponses ainsi que le traitement des informations. Ces objets au nombre de cinq sont énoncés par Goyette :

- Objet Application
  L'objet Application sert à partager des informations entre tous les utilisateurs d'une application donnée.
- Objet Request
  L'objet Request sert à accéder à toute information transmise dans une requête HTTP.
- Objet Response
  L'objet Response sert à contrôler les informations envoyées à l'utilisateur.
- Objet Server
  L'objet Server permet d'accéder aux méthodes et aux propriétés du serveur.
- Objet Session
  Appelle les méthodes et propriétés de l'objet Session. [GOYETTE99]

Notons encore qu'ASP étant un standard Microsoft, ces fichiers fonctionnent avec les serveurs WEB suivants : Internet Information Server et Personnal Web Server de Microsoft. Mais du fait que les pages ASP s'exécutent sur le serveur, les applications qui les intègrent sont accessibles depuis n'importe quel navigateur, quelle que soit la plate-forme. La renommée acquise par ASP vis-à-vis des programmeurs et des développeurs, ainsi que les évolutions que cette technologie a connues permettent à ASP de tourner maintenant sur plusieurs plates-formes et plusieurs systèmes d'exploitation.

1.4. ODBC

Open Data Base Connectivity (ODBC) est un protocole défini par Microsoft. Il permet la communication entre des applications clientes fonctionnant sous Windows et la plupart des systèmes de gestion de base de données les plus courants. Le protocole ODBC permet d'interfacer une application cliente avec un serveur de base de données possédant un driver ODBC. La plupart des systèmes de gestion de base de données en possède un. ODBC offre donc à toutes ces applications clientes un langage commun.

Ainsi, « Les applications compatibles ODBC (souvent dénommées 'applications-client') envoient des requêtes à d'autres applications compatibles ('applications serveur

ODBC') au travers du gestionnaire de pilotes ODBC, situé sur la machine cliente. Le gestionnaire de pilotes ODBC détermine quelle est la source de données utilisée et quel pilote ODBC peut communiquer avec cette source particulière. La requête est alors envoyée via le pilote au serveur ODBC - habituellement une base de données. La base de données peut être locale, sur le même ordinateur ou distante. Les données résultant de la requête sont alors renvoyées via le gestionnaire de pilotes ODBC à l'application-client. Le langage ODBC est une combinaison d'appels aux fonctions de l'API ODBC et du langage d'interrogation SQL. ». [FILEMAKER01]

Le principal inconvénient de ce protocole est qu'il reste une propriété exclusive de Microsoft. Dès lors, au même titre qu'Internet Information Server, cette technologie ne fonctionne que sous système d'exploitation Windows.

2. Client-Side

De son côté, le client doit tout simplement être équipé d'un navigateur ou *browser* en anglais. Il existe différents types de navigateurs mais les deux plus fréquents sont Netscape Navigator et Internet Explorer. Ces deux navigateurs sont équipés d'une DTD (Definition Type Document) pour comprendre le langage HTML. Nous détaillerons dans le chapitre suivant le fonctionnement d'une DTD.

Nous n'allons pas nous attarder plus longtemps dans cette section, les navigateurs étant maintenant bien connu de chacun. Notons simplement que, généralement, les navigateurs sont suffisants pour consulter l'information sur Internet. Mais dans certaines situations on doit leur ajouter ce qu'on appelle des 'plugs-in' tels que RealPlayer pour écouter différents sons sur Internet ou encore QuickTime pour ce qui est des vidéos.

Conclusion de chapitre

Il faut retenir que le choix des technologies nous a été imposé. Le système de gestion de bases de données ORACLE nous paraît un choix judicieux par sa simplicité et sa large diffusion. De plus, le langage SQL est simple et facile d'utilisation.

En ce qui concerne le serveur WEB, Personnal Web Server de Microsoft, le choix paraît lui aussi pertinent car l'application est très simple à utiliser. Mais l'utilisation de ce serveur restreint le champ du prototype. A terme, le choix du serveur WEB reste incertain et ouvert.

Le problème de compatibilité apparaît pour la technologie Active Server Page, étant donné que celle-ci appartient à Microsoft. Mais aujourd'hui, sa flexibilité et sa simplicité d'utilisation lui permettent de s'étendre à d'autres systèmes d'exploitation et à d'autres serveurs WEB. Le protocole ODBC suit le même chemin car il est largement répandu et est présent dans la plupart des systèmes de gestion de base de données. Ces deux technologies nous paraissent donc un choix judicieux.

Enfin, du côté client, la seule utilisation d'un navigateur simplifie grandement la vie des utilisateurs.

Nous aborderons dans le chapitre suivant, les différents langages de programmation utilisé pour la création du prototype. Nous verrons ainsi les langages HTML, XML, XSL, Jscript ainsi que l'interface ADO.

Cinquième chapitre

Langages utilisés dans le prototype Anthémis

I. Le langage HTML
   1. Mise en situation
   2. Description

II. Le langage XML
   1. Mise en situation
   2. Description

III. Le langage XSLT
   1. Les feuilles de style
      1.1. Mise en situation
      1.2. Description
   2. Le langage XSLT
      2.1. Mise en situation
      2.2. Description
         2.2.1. xsl:stylesheet
         2.2.2. xsl:template
         2.2.3. xsl:apply-templates
         2.2.4. xsl:value-of

IV. Le langage Jscript
   1. Mise en situation
   2. Description

V. L'interface ADO
   1. Mise en situation
   2. Description
      2.1. L'objet Connection
         2.1.1. La méthode Open()
         2.1.2. La méthode Execute()
         2.1.3. La méthode Close()
      2.2. L'objet Recordset
         2.2.1. La méthode Open()
         2.2.2. La méthode MoveNext()
         2.2.3. La méthode Close()

Introduction de chapitre

La création du prototype nécessitait l'utilisation de plusieurs langages. Nous allons donc dans ce chapitre réaliser une rapide explication de ceux-ci afin de permettre aux lecteurs de comprendre plus aisément la suite de ce travail. L'explication du langage Jscript que nous donnerons sera moins importante que les explications concernant les autres langages pour la simple raison que ce langage ressemble en certains points au langage JAVA que nous avons étudié aux facultés. Nous détaillerons plus précisément les langages XML et XSL qui étaient, pour nous, des matières nouvelles.

Avant tout, nous avons dû apprendre le langage HTML qui n'est plus à présenter dans les détails avec l'émergence de l'Internet. Nous avons commencé par apprendre ce langage car il est en quelque sorte à la base des langages XML et XSL. Le langage HTML est le plus répandu, il est très simple d'utilisation et ne nécessite pas de nombreux pré-requis.

Une fois ce langage maîtrisé, nous avons débuté l'apprentissage du langage XML. Le langage XML est très proche du langage HTML mais n'est pas aussi rigide que ce dernier et propose ainsi une plus grande liberté au programmeur. En effet, le programmeur convient lui-même de ses propres balises.

Juste après le langage XML, nous nous sommes penchés sur le langage XSL, celui-ci permettant de mettre en forme les documents XML. Le langage XSL présente, comme nous le verrons, un sous-langage de transformation XSLT. Ce dernier permet de transformer les données contenues dans le langage XML dans un autre document XML afin de les présenter.

Une fois ces trois langages maîtrisés, nous avons étudié le langage Jscript qui est très répandu actuellement. Ce dernier permet comme son nom l'indique d'effectuer des scripts. Ces scripts peuvent être insérés dans des documents HTML ainsi que dans des documents ASP.

Pour terminer, nous nous sommes intéressés à l'interface ADO qui permet d'assurer, quant à elle, des relations avec une base de données.

Tous ces langages sont très utilisés à l'heure actuelle et méritent d'être brièvement décrits. Ces langages sont bien adaptés à la gestion de base de données et à l'interaction de celle-ci avec le prototype. De plus, le choix de ces langages s'est opéré de manière à rendre simple et efficace la diffusion et la présentation de la connaissance.

I. Le langage HTML

Le langage HTML[3] (Hyper Text Markup Language) est un langage de programmation à balises (tags en anglais). Ce langage a été créé en 1992 et a depuis évolué sous plusieurs versions. Nous pouvons retrouver la description de ces différentes versions sur la page du W3C (World Wide Web Consortium). La version que nous avons utilisée et dont nous ferons une brève synthèse dans ce présent travail est la quatrième version : HTML 4. [W3C98]

1. Mise en situation

Ce langage est le langage en rigueur sur le WEB. Il est simple à utiliser (un éditeur de texte suffit pour créer des pages HTML) et est reconnu par l'ensemble des navigateurs. Un des aspects les plus importants et les plus attrayants de ce langage est la possibilité de faire des liens entre des pages différentes par le mécanisme des liens hypertextes.

[3] Pour une explication plus approfondie des différentes balises et des attributs correspondants à ces balises nous vous conseillons de lire la deuxième partie du livre *'Programmation Internet, HTML 4,XML et JAVA2. Ressources d'experts.'* écrit par Ladd [LADD00]

Une page HTML est donc définie comme un ensemble d'éléments. Pour identifier un élément, on utilise une balise qui se définit comme un marqueur de début de l'élément et un marqueur de fin de l'élément. Les marqueurs de début sont de la forme **<balise>** et les marqueurs de fin de la forme **</balise>**. L'énoncé de la balise est identique dans le marqueur de début et de fin. Une balise peut avoir des attributs pour spécifier certaines caractéristiques de présentation ou de mise en pages.

2. Description

Il existe plusieurs balises définies dans la recommandation HTML 4. Nous pouvons regrouper toutes ces balises dans différentes classes. Nous ne verrons pas dans ce travail l'explication de toutes les balises mais nous allons répertorier les plus utilisées dans le tableau suivant. (TABLEAU 5.1) Il faut également savoir qu'il existe deux sortes de balises. Les balises conteneurs doivent avoir une balise ouvrante et une balise fermante. Les balises autonomes ne doivent pas avoir de balises fermantes.

Classe de balise	Exemples	Description
Balises de structure	<HTML>, <HEAD>, <BODY>, <SCRIPT>, <TITLE>, <LINK>	Une page HTML est composée de trois membres principaux. La déclaration HTML, l'en-tête (head) et le corps (body).
Balises de formatage	<B>, <I>, <U>, , <CENTER>, <HR>, <H1>...<H6>, <P>	Les balises de formatage offrent des possibilités de jouer sur la présentation du contenu de la page.
Balises de listes	<LI>, <OL>, <UL>, <DL>, <DT>, <DD>, <MENU>, <DIR>	Les balises de listes permettent une énumération des éléments sous la forme d'une liste. Il en existe cinq sortes.
Balises d'hyperliens	<A>	Les balises d'hyperliens permettent de passer d'une page à une autre en un seul click de souris.
Balises d'images	<IMG>, <MAP>, <AREA>	Les balises images permettent de remplir une page avec une image en plus d'un texte.

Balises de tableaux	<TABLE>, <COL>, <TR>, <TD>, <TH>	Les balises tableaux permettent d'introduire des tableaux dans la page.
Balises de formulaires	<FORM>, <INPUT>, <LABEL>, <BUTTON>, <TEXTAREA>	Les balises de formulaires s'occupent de gérer un formulaire qui sera en connexion avec un serveur.
Balises de cadres	<FRAMESET>, <FRAME>	Les balises de cadres permettent de diviser une page en plusieurs cadres autonomes.
Balises de contenus exécutables	<APPLET>, <PARAM>, <OBJECT>	Les balises de contenus exécutables permettent d'exécuter et de produire un contenu dynamique.

TABLEAU 5.1 : Classification des types de balises d'HTML 4.

Nous remarquons donc que le langage HTML est doté de plusieurs balises qui ont chacune soit un rôle de présentation, soit un rôle de structuration de page. Si nous prenons la balise **<I>** et que nous écrivons dans une page HTML : <I> Bonjour tout le monde <I>, le résultat attendu est la phrase : *Bonjour tout le monde*.

Ces balises peuvent, elles-mêmes, être accompagnées d'attributs qui modifieront leur comportement. Certains attributs sont indispensables. Une balise peut recevoir plusieurs attributs ou aucun. La balise **<BODY>** peut ainsi être accompagnée de l'attribut **TEXT = « yellow »** qui signifie que le texte compris entre la balise ouvrante **<BODY>** et la balise fermante **</BODY>** sera en jaune. Nous noterons ce désir comme suit :
<BODY TEXT = « yellow »> … </BODY>

La structure générale d'une page HTML (SCHEMA 5.1), se présente habituellement selon la structure suivante. Toute page HTML commence logiquement avec la balise <HTML>. Il est toutefois possible de retrouver, en première position d'une page HTML, la balise <!doctype> qui indique avec quelle version d'HTML l'on travaille. Au sein de la balise <HTML>, on retrouve la balise <HEAD> et la balise <BODY>. La balise <HEAD> contient de l'information sur la page elle-même et spécifie les liens généraux vers d'autres pages. La balise <BODY> forme le corps, elle contient donc les éléments de texte, d'images, de liens hypertextes, de tableaux, etc.

```
<HTML>
        <HEAD>
        ...
        < /HEAD>
        <BODY>
        ...
        </BODY>
</HTML>
```

SCHEMA 5.1 : Structure formelle d'un document HTML.

II. Le langage XML

XML[4] (eXtended Markup Language) est lui aussi un langage de balises. Il a été créé en 1996 et a la particularité de travailler avec des balises qui ne sont pas prédéfinies à l'opposé de HTML. Le W3C a publié sa recommandation en octobre 2000. [W3C00]

1. Mise en situation

Au départ, nous disposions du SGML (Standard General Markup Language), créé en 1989, lui aussi est un langage travaillant avec des balises. Ce langage est extrêmement flexible et permet de formaliser toutes formes de pages WEB. Le problème rencontré avec ce langage est sa surqualification pour l'ensemble des publications du WEB. En bref, le SGML est bien trop compliqué pour les besoins des individus travaillant sur le WEB.

Quelques années plus tard le HTML a vu le jour et comme vu précédemment, celui-ci s'est fortement répandu sur le WEB. Mais il n'est pas exempt de limites. En effet le HTML rencontre deux problèmes majeurs. Le premier problème se retrouve dans le fait que le HTML est trop restrictif et rend donc la description de pages complexes très

[4] Pour une explication plus approfondie des différentes balises et des attributs correspondants à ces balises, nous vous recommandons la troisième partie du livre *'Programmation Internet, HTML 4, XML et JAVA 2. Ressources d'experts.'* . [LADD00] Nous vous conseillons également de lire la recommandation sur le langage XML publié par le W3C. [W3C00]

difficile. Le second problème vient du fait que HTML est de plus en plus orienté vers la présentation du contenu plutôt que vers la description de celui-ci.

Plus tard, le langage XML a été développé, celui-ci se présente comme un compromis idéal entre le HTML et le SGML. En effet, nous avons d'un côté le HTML qui est un langage simple mais qui apparaît comme un langage de présentation de données, et de l'autre côté, nous avons le SGML qui est un langage de structuration de données mais qui est très complexe. XML est, quant à lui, exclusivement destiné à la description du contenu. Il s'agit en outre d'un langage extrêmement flexible.

En résumé, pour aboutir au XML, le SGML a été épuré de toutes les caractéristiques inappropriées à la publication WEB. Il en résulte un métalangage (langage qui permet de définir d'autres langages) qui hérite de la structure et de la flexibilité du SGML, la complexité en moins. Nous allons reprendre dans le TABLEAU 5.2 les points forts du langage XML.

Flexible	Nous pouvons introduire dans le langage nos propres balises XML, adaptées à nos besoins.
Portable	Le XML nous permet de rassembler dans un fichier les règles qui régissent notre formatage, afin que nos pages soient lisibles et manipulables par d'autres.
Structuré	Le XML a hérité du SGML le respect très strict de la structure. Une page qui n'est pas correctement structurée ne sera pas considérée comme une page XML.
Descriptif	Les éléments XML servent exclusivement à décrire la signification de leur contenu.

TABLEAU 5.2 : Particularité du langage XML.

2. Description

Il existe en XML cinq types de balisage pour former une page XML. Nous allons reprendre ces différents types dans le TABLEAU 5.3. Nous y énoncerons donc les différents types, nous les décrirons et nous en donnerons la syntaxe.

Types	Description	Syntaxe
Eléments	Décrivent le sens du contenu des balises. Il existe des éléments vides.	<NOM>… </NOM> ou (élément vide)
Entités	Permettent de représenter des caractères réservés.	Caractère '<' → '&lt ;' Caractère '>' → '&gt ;'
Comments	Permet d'insérer des commentaires dans le code.	< !-- … -->
Instructions de traitement	Permettent d'incorporer des informations qui seront transmises directement à partir de la page.	< ?xml version = '1.0' ?>
Sections ignorées	Permettent de passer à une application des caractères réservés XML.	< ![CDATA[4 < 3 is FALSE.]]>

TABLEAU 5.3 : Classification des types de balises de XML.

Nous allons maintenant illustrer nos propos par un exemple simple d'un texte balisé par XML. (SCHEMA 5.2)

```
<LETTRE>

    <DATE ALIGN = 'RIGHT'>
    18 juillet 2001
    </DATE>

    <ADRESSE>
    Roc & Cos <BR/>
    38 rue des Haies <BR/>
    6032 Mont sur Marchienne <BR/>
    </ADRESSE>

    <DESTINATAIRE>
    Cher responsable de la clientèle,
    </DESTINATAIRE>
```

```
<CORPS ALIGN = 'JUSTIFY'>
Nous voudrions commander trois ordinateurs de type XY.
</CORPS>

<SALUTATIONS>
Veuillez agréer, monsieur, nos salutations distinguées.
</SALUTATIONS>

<SIGNATURE>
Jean-Philippe Rochet.
Alexis Cossement.
</SIGNATURE>

</LETTRE>
```

SCHEMA 5.2 : Illustration d'une lettre en XML.

A la lecture de ce fichier XML, le sens du balisage nous apparaît clairement et il le sera sans doute également pour d'autres personnes lisant ce fichier. Par contre, les analyseurs XML, quant à eux, ne comprendront pas ce fichier. C'est pourquoi nous devons l'accompagner d'une DTD (Définition de Type de Document) qui spécifiera nos éléments, leurs attributs et leur syntaxe. Ainsi tous les analyseurs XML pourront interpréter correctement le contenu de notre fichier XML.

Nous allons donc maintenant voir en quoi consiste une DTD. Une DTD décrit l'ensemble des règles qui régissent le balisage d'une page XML. On y définit chaque élément, les attributs que cet élément est susceptible de recevoir et les valeurs que peuvent prendre ces attributs ainsi que les éléments qui peuvent être contenus dans d'autres éléments.

Nous allons reprendre dans le SCHEMA 5.3 la DTD correspondant à l'illustration de la lettre décrite ci-dessus dans un formalisme XML.

```
< !ELEMENT lettre (date, adresse, destinataire, corps, salutations, signature ?) >
< !ELEMENT date (#PCDATA) >
< !ATTLIST date align (left | right) 'left' >
< !ELEMENT adresse (#PCDATA | br*) >
< !ELEMENT br EMPTY >
< !ELEMENT destinataire (#PCDATA) >
< !ELEMENT corps (#PCDATA) >
< !ATTLIST corps align (left | justify | right) 'left' >
< !ELEMENT salutations (#PCDATA) >
< !ELEMENT signature (#PCDATA) >
```

SCHEMA 5.3 : DTD en correspondance avec le SCHEMA 5.2.

Cette DTD signifie que l'élément **lettre** doit contenir les éléments 'date', 'adresse', 'destinataire', 'corps', 'salutations' et facultativement l'élément 'signature' (présence du point d'interrogation pour cet élément).

L'élément **date** contient uniquement du texte. Il peut être accompagné de l'attribut **align** qui prend les valeurs 'left' ou 'right'. Par défaut si l'attribut **align** n'est pas spécifié, l'élément **date** prend la valeur 'left'.

L'élément **adresse** peut contenir du texte, ainsi que des éléments <BR/>. L'élément **br**, quant à lui, est un élément vide, c'est-à-dire qu'il n'est pas pourvu de balise fermante.

Les éléments **destinataire, corps, salutations, signature** contiennent tous les quatre uniquement du texte. L'élément **corps**, quant à lui, peut être accompagné de l'attribut **align** qui prend les valeurs 'left' ou 'right' ou 'justify'. Par défaut si l'attribut n'est pas donné, la valeur est 'left'.

Une fois la DTD écrite, il faut spécifier à l'analyseur l'emplacement de la DTD du fichier XML concerné. Deux solutions sont possibles, soit nous plaçons la DTD au sein du fichier XML : < !DOCTYPE LETTRE [...] >, soit nous précisons dans le fichier XML l'emplacement de la DTD correspondante :

< !DOCTYPE LETTRE SYSTEM 'http://www.serveur.com/dtd/lettre.dtd' >

Dans XML, toute personne peut donc définir ses propres balises ainsi que leur agencement et la structure de celles-ci dans une DTD écrite par ses soins. Une page XML peut dès lors être soumise à un ensemble de contraintes réunies dans la DTD. Ces contraintes sont : fixer le nom des balises, fixer l'ordre de ces balises, fixer les attributs attachés à celles-ci. Cependant, la présence d'une DTD n'est pas obligatoire. Lorsqu'une page XML est conforme aux spécifications de la syntaxe XML (à chaque balise ouvrante correspond une balise fermante), on la qualifie de bien formée. Lorsqu'une page XML est bien formée et qu'elle vérifie les contraintes d'une DTD, elle est considérée comme valide par rapport à cette DTD. Il existe dans les nouvelles versions des navigateurs un parseur pour vérifier la conformité et la validation d'une page XML.

Contrairement à HTML, une page XML ne contient aucune information sur la présentation de son contenu. Cette présentation devra donc être dictée par l'utilisation de feuilles de style ou par l'utilisation du langage XSL (Extensible Stylesheet Language).

En pratique, une page XML doit toujours avoir une balise racine unique au sein de laquelle se trouvera toutes les autres balises. Une page XML doit commencer par une ligne de code qui permet de l'identifier comme étant une page XML. Un fichier XML prend l'extension *.xml.

III. Le langage XSLT

Le langage XSLT (Extensible Stylesheet Language Transformation) est une partie du langage XSL qui est un langage de présentation propre à XML au même titre que les feuilles de style pour le langage HTML. Pour bien comprendre le but d'un langage de présentation, nous allons présenter brièvement le but des feuilles de style du langage HTML.

1. Les feuilles de style

Nous avons vu que le langage HTML avait pour fonction principale de présenter le contenu et non de le décrire. Pour remédier à ce problème le W3C a publié la directive CSS1 (Cascading Style Sheets 1) qui tend à formaliser les instructions de présentation. En 1998, la directive CSS2 (Cascading Style Sheets 2) a été rendue publique. Celle-ci apporte quelques modifications à la première directive, comme la gestion de la transcription vocale.

1.1. Mise en situation

Qu'est ce qu'une feuille de style[5] ? Une feuille de style définit l'ensemble des propriétés de mise en forme qui doivent être appliquées au texte. Ces propriétés concernent les attributs de police, tels que la forme d'écriture, le style (gras, italique, souligné, etc.), la couleur, l'alignement. La feuille de style donne également des mesures précises quant aux marges, à l'interlignage, etc.

Pourquoi recourir à des feuilles de style ? Celles-ci garantissent un contrôle précis de la présentation. En effet, elles permettent **de centraliser les propriétés de style** en stockant ces propriétés dans un fichier unique et en les appliquant à toutes les pages. Dès lors, des modifications de présentation affectant l'ensemble du site peuvent être réalisées en une seule fois. En plus, elles permettent **de respecter la mise en forme** d'un navigateur à l'autre en intégrant dans ces feuilles des propriétés de style précises.

1.2. Description

La directive CSS2 autorise trois techniques pour inclure les définitions de style dans un document. Il est évidemment possible de combiner ces trois techniques que nous

[5] Pour une explication plus approfondie de l'utilisation des propriétés des feuilles de styles, nous vous renvoyons au chapitre 9 du livre *'Programmation Internet, HTML 4, XML et JAVA 2. Ressources d'experts'* [LADD00]

détaillerons par après mais il faut faire attention en combinant ces techniques à ne pas négliger la cohérence de la présentation.

La première technique est appelée **styles liés**. Cette technique permet de lire l'information de présentation depuis un fichier extérieur dont l'URL est spécifiée dans la balise <LINK> de la page HTML. Pour ce faire, il faut créer un document contenant l'ensemble des informations de style. Ce document prendra l'extension *.css et aura comme syntaxe : X { propriété : valeur [; propriété : valeur]* } où X est le nom d'une balise de la feuille HTML. (SCHEMA 5.4) Du côté de la page HTML, nous devons retrouver, dans l'en-tête de celle-ci, la balise <LINK> dans laquelle nous donnons l'emplacement de la feuille de style :
<LINK REL = STYLESHEET HREF = 'feuilledestyle.css'>

```
BODY { font : 14 pt Times New Roman ; color : red }
H1 { font : 18 pt Arial ; color : blue }
P { font : 14 pt Times New Roman ; line-height : 16 pt }
```

SCHEMA 5.4 : Exemples de syntaxe d'une feuille de style.

La seconde technique, appelée **styles incorporés**, définit la feuille de style au sein de l'en-tête de la page HTML grâce à la balise <STYLE>. La syntaxe des styles définis est la même que pour la technique des styles liés. Nous devons donc dans l'en-tête de la page HTML écrire : <STYLE TYPE='TEXT/CSS'>...</STYLE>

La dernière technique, appelée **styles intégrés**, permet de définir le style au sein des balises d'une page HTML. Le style s'applique alors au contenu de la balise. On doit alors, dans la balise, ajouter l'attribut STYLE qui définira le style de présentation du contenu de la balise.

2. Le langage XSLT

Le langage XSL est un langage de mise en forme de documents XML. Il est composé de deux sous-langages qui sont un langage de transformation et un langage de présentation. La partie langage de transformation (XSLT[6]) a été isolée et redéfinie par le W3C dans la recommandation de novembre 1999. [W3C99] C'est donc grâce à XSL que peuvent être réalisées des présentations de document XML, que celles-ci soient liées au WAP, au WEB ou à tout autre support électronique.

2.1. Mise en situation

Nous avons vu précédemment que XML permet de déclarer des balises et des données à l'aide d'éléments définis par l'auteur du fichier tout en spécifiant leur nature. Il en résulte une structure de données personnalisable et mieux organisée qu'en HTML. Mais pour que les données à l'intérieur d'un élément soient parfaitement interprétées, elles doivent être affichées de sorte que le lecteur puisse instinctivement et immédiatement en discerner l'objet et l'utilité. Un style doit donc être donné à l'élément. XSL fait en sorte que les données d'un élément soient affichées de la façon souhaitée par l'auteur.

XSL inclut un vocabulaire XML pour la spécification de formatage et un mécanisme XSLT de transformation d'un document XML en un autre document XML. XSL spécifie les règles de présentation d'un document XML en utilisant XSLT pour décrire comment le document peut être transformé en un autre document qui utilise le vocabulaire de formatage. XSLT est aussi conçu pour être utilisé indépendamment de XSL. Cependant, XSLT n'est pas censé être utilisé comme un langage de transformation XML à vocation générale. Il a surtout été conçu pour les types de transformations nécessaires lorsque XSLT est utilisé comme une partie de XSL.

---

[6]Pour une explication plus approfondie de l'utilisation de XSLT, nous vous conseillons de lire le cours *'Développement de sites WEB transactionnels'* donné par le Professeur Gerbé au HEC de Montréal. http://tim.hec.ca/olivier.gerbe/cours/display.asp?xml=4-72100&xsl=live_seance&no=12. Ainsi que la recommandation XSLT publiée par le W3C. [W3C99]

2.2. Description

Il faut savoir que le document XML est transformé, dans la mémoire de l'ordinateur, en un arbre XML. Le langage XSLT va pouvoir sélectionner les différents nœuds de l'arbre XML grâce au langage XPATH (XML PATH language). Or, un document XSLT reprend un ensemble de gabarits (templates en anglais). Ceux-ci seront appliqués sur les nœuds de l'arbre source XML afin de générer un document de sortie. Un gabarit est composé d'éléments permettant de créer des résultats dans le document de sortie ainsi que d'éléments permettant de sélectionner des nœuds et de leur appliquer un autre gabarit.

Nous allons maintenant reprendre les différents éléments les plus souvent utilisés dans un document XSLT. L'explication de ces différents éléments a été reprise des transparents du cours donné en 2000-2001 par le Professeur Gerbé à l'école des Hautes Etudes Commerciales de Montréal, cours intitulé *'Développement de sites WEB transactionnels'*.

2.2.1. xsl:stylesheet

L'élément **xsl:stylesheet** de la feuille de style XSLT contient tous les autres éléments de la feuille de style.

Syntaxe : **<xsl:stylesheet** version="1.0"
Xmlns:xsl = "http://www.w3.org/TR/WD-xsl">
</xsl:stylesheet>
où xmlns:xsl = "http://www.w3.org/TR/WD-xsl" définit l'espace de nom utilisé dans le document.

2.2.2. xsl:template

L'élément **xsl:template** permet de définir un gabarit qui sera appliqué à tous les éléments du document XML qui correspondent au chemin d'accès.

Syntaxe : **<xsl:template match** = "*chemin*" >

. . .

</xsl:template>

où *chemin* est un chemin d'accès XPATH.

Le contenu de l'élément **<xsl:template>** comprend ce qui va être exécuté quand les nœuds sélectionnés vont correspondre au chemin.

2.2.3. xsl:apply-templates

L'élément **xsl:apply-templates** permet de sélectionner les nœuds et de leur appliquer des gabarits.

Syntaxe : **<xsl:apply-templates select** = "*chemin*"/>

où *chemin* est un chemin d'accès XPATH.

Le processeur XSLT sélectionne les nœuds qui correspondent au chemin et cherche les gabarits applicables.

2.2.4. xsl:value-of

L'élément **xsl:value-of** permet d'obtenir un résultat (un texte) à partir de données du document source.

Syntaxe : <**xsl:value-of** select = "*chemin*"/>

où *chemin* est le chemin d'accès à l'élément dont on désire le résultat.

IV. Le langage Jscript

Le langage Jscript[7] est un langage de programmation orienté objet créé par Microsoft. Il correspond au langage JavaScript développé par Netscape. Ces deux langages sont logiquement compatibles même s'il existe quelques différences.

1. Mise en situation

Jscript est un langage interprété, c'est-à-dire que l'ordinateur doit analysé le programme chaque fois qu'il est exécuté. Un script peut être inséré dans une page HTML ou écrit dans un fichier extérieur au document HTML. Dans ce dernier cas, il est fourni au document HTML par une commande appropriée. Une fois les lignes Jscript incorporées dans une page HTML ou fournies dans un autre document, tout navigateur qui sait gérer Jscript pourra interpréter ces lignes et les exécuter.

Pourquoi utiliser des scripts ? Les pages HTML sont statiques. Ainsi, après leur création, les pages HTML ne permettent l'interaction avec l'utilisateur que par le biais de liens hypertextes. Par contre avec l'emploi de script, les pages HTML deviennent plus interactives. JavaScript permet d'écrire des scripts qui s'exécuteront sur le navigateur du client plutôt que sur le serveur, c'est ce que l'on appelle 'le Client-side scripting'. Jscript, quant à lui, permet d'exécuter un script sur le serveur avant l'envoi de la page HTML vers

[7] Pour de plus amples informations sur le langage JavaScript, nous vous conseillons de lire la quatrième partie du livre *'Programmation Internet, HTML 4, XML et JAVA 2. Ressources d'experts'*. [LADD00]

le client, c'est ce que l'on appelle 'le Server-side scripting'. Ce dernier type de script est utilisé pour les pages ASP.

2. Description

Nous retrouvons les scripts au sein des balises <SCRIPT> et </SCRIPT> d'une page HTML. Cette balise est accompagnée de l'attribut LANGUAGE qui définit le langage de script employé (soit JavaScript, soit Jscript, soit VBScript) pour évaluer le script et SRC qui peut être employé pour charger un script à partir d'une source externe.

Il est possible de placer son script n'importe où dans la page HTML mais il est conseillé de le placer au début de la page au sein des balises <HEAD>. En effet, vu qu'un script peut être employé de n'importe quel élément de la page HTML et que le navigateur analyse une page HTML de haut en bas, si le script est placé au début de la page alors on est certain qu'il sera analysé avant qu'un élément ne l'appelle.

Nous pouvons aussi retrouver le langage Jscript au sein des balises <% ... %>. C'est le cas pour notre prototype Anthémis. Ces balises correspondent à la délimitation d'un fichier ASP qui est du même gabarit qu'un fichier HTML, à part qu'il est possible qu'aucune balise HTML ne figure dans un fichier ASP. Nous retrouverons au sein de ces balises toutes les lignes de code qui permettent d'interagir avec la base de données. La balise d'introduction est notée : <% @language = JScript %> comme nous l'avons vu dans le chapitre précédent. Il est vivement conseillé par Microsoft d'utiliser avec les pages ASP les scripts de types Jscript ou VBScript pour une question de conformité.

III. L'interface ADO

L'interface ADO (ActiveX Data Objects) offre un moyen d'accéder à l'ensemble des sources de données, et ce, quel que soit le langage utilisé.

1. Mise en situation

Pour assurer l'interface avec une base de données, il est nécessaire d'utiliser des outils de programmation pour accéder à la structure de la base de données et à ses données. Ainsi, on doit pouvoir récupérer le schéma d'une table; ajouter, modifier et supprimer un enregistrement; naviguer à l'intérieur d'une table; faire une requête, etc. Toutes ces tâches seront réalisées à l'aide des ADO de Microsoft. Il faut donc rendre possible six phases :

- la connexion à une base de données,
- la spécification de la requête à exécuter,
- l'exécution de la requête,
- la lecture des résultats,
- le traitement des résultats,
- la déconnexion à la base de données.

2. Description

En ce qui concerne l'implémentation de notre prototype, nous avons utilisé deux objets d'ADO. L'objet **Connection** et l'objet **RecordSet**.[8]

2.1. L'objet Connection

L'objet **Connection** est l'objet de plus haut niveau dans la librairie d'ADO. Ces principales méthodes sont *Open()*, *Execute()* et *Close()*. La création d'un objet de type **Connection** se fait en utilisant la méthode *CreateObject()* de l'objet **Server**.

[8] La suite de cette partie se base sur le cours *'Développement de sites WEB transactionnels'* donné par le Professeur Gerbé au HEC de Montréal. http://tim.hec.ca/olivier.gerbe/cours/display.asp?xml=4-721-00&xsl=live_seance&no=10

Syntaxe : Connexion = Server.**CreateObject**("ADODB.**Connection**");

2.1.1. La méthode Open()

La méthode *Open()* permet d'ouvrir une connexion à une base de données.

Syntaxe : Connexion.**Open** ("DSN=bde; uid=demo; pwd=demo");
Paramètres : On retrouve en paramètre une chaîne de caractères qui contient l'information de la connexion, c'est-à-dire le nom de la base de données, le nom de l'utilisateur et son mot de passe.

2.1.2. La méthode Execute()

La méthode *Execute()* permet d'envoyer une requête au SGBD. La requête est une requête SQL étant donné que le SGBD est dans notre cas ORACLE.

Syntaxe : 1) Envoi d'une requête sans résultat :
Connexion.**Execute** (ReqSQL);
2) Envoi d'une requête avec résultats, ce qui retourne un objet **RecordSet** :
résultat = Connexion.**Execute** (ReqSQL);
avec ReqSQL, une requête SQL du style : "SELECT * FROM commande";

2.1.3. La méthode close()

La méthode *close()* permet tout simplement de fermer une connexion.

Syntaxe : Connexion.**close**();

2.2. L'objet Recordset

L'objet **RecordSet** représente l'ensemble des enregistrements obtenus à partir d'une requête. Ces principales méthodes sont *Open()*, *MoveNext()* et *Close()*. La création d'un objet de type **RecordSet** se fait en utilisant la méthode *CreateObject()* de l'objet **Server**.

Syntaxe : resultatSQL = Server.**CreateObject** ("ADODB.**RecordSet**");

2.2.1. La méthode Open()

La méthode *Open()* permet d'exécuter une requête SQL et de poser le pointeur sur la première ligne.

```
Syntaxe :   reqSQL = "SELECT code FROM commande";
            resultatSQL.Open (reqSQL, Connexion);
```

2.2.2. La méthode MoveNext()

La méthode *MoveNext()* permet de déplacer le pointeur sur l'enregistrement suivant. Il est souvent utilisé dans une boucle où l'on teste la propriété EOF (End of File).

Celle-ci permet de savoir si la position du pointeur se trouve après le dernier enregistrement du **RecordSet**.

La propriété EOF retourne la valeur "true" si la position du pointeur se trouve après le dernier enregistrement ou la valeur "false" si la position du pointeur se trouve sur le dernier enregistrement ou avant celui-ci .

```
Syntaxe : while ( ! resultatSQL.EOF)
          { ...
            resultatSQL.MoveNext();
           }
```

2.2.3. La méthode close()

La méthode *close()* permet de fermer une requête.

```
Syntaxe : resultatSQL.close();
```

Conclusion de chapitre

En conclusion, les langages utilisés permettent bien de rencontrer l'objectif du prototype. En effet, le langage HTML permet de construire des pages WEB de façon simple et efficace. Quant au langage XML couplé au langage XSLT, il permet de générer automatiquement l'approvisionnement de la base de données et laisse une grande liberté dans la présentation des pages et des connaissances. De plus, il permet de fixer un certain standard et une certaine uniformité à travers ces pages. Nous verrons dans les chapitres

ultérieurs comment ces deux langages permettent d'insérer les connaissances dans la base de données.

Le langage Jscript permet quant à lui d'exécuter des scripts à l'intérieur des pages HTML par leur simple insertion à l'intérieur de celles-ci. De plus, il permet aussi leur exécution en dehors des pages HTML par l'utilisation d'une simple ligne de commande. Mais surtout ce langage permet l'interaction avec les pages ASP. Enfin, Le langage ADO qui permet d'assurer les interactions avec la base de données est lui aussi très simple d'utilisation et garantit la connexion avec la base de données.

Nous allons maintenant développer dans les deux chapitres suivant l'implémentation du prototype de gestion des connaissances. Dans le sixième chapitre, nous aborderons l'implémentation du prototype par le biais des modèles Entité-Association et dans le septième, nous détaillerons l'implémentation par le biais des graphes conceptuels.

Sixième chapitre

Modèle Entité-Association pour le prototype Anthémis

I. Formalisme des bases de données relationnelles

1. Concepts des bases de données relationnelles
2. Modèle Entité-Association

II. Prototype de gestion de la connaissance par le modèle Entité-Association

1. Architecture de la base de données
   1.1. Le niveau modèle
   1.2. Le niveau connaissance
2. Introduction du modèle et des connaissances
   2.1. Introduction du modèle
      2.1.1. Formalisme XML
      2.1.2. Agencement des balises
      2.1.3. Création d'une DTD pour le modèle
   2.2. Introduction des connaissances
      2.2.1. Formalisme XML
      2.2.2. Agencement des balises
      2.2.3. Etude de cas
3. Intervention des fichiers XSL
   3.1. Fichier XSL pour le modèle
   3.2. Fichier XSL pour les connaissances
4. Evolutions du prototype Anthémis

Introduction de chapitre

Dans ce chapitre, nous allons présenter l'implémentation du prototype Anthémis par le biais du modèle Entité-Association. Pour cela, nous allons diviser ce chapitre en deux parties. La première nous expliquera le principe des modèles Entité-Association et la deuxième nous montrera comment le prototype fonctionne via ce principe.

Nous parlerons donc, dans un premier lieu, du formalisme des bases de données relationnelles. Dans cette première partie, nous détaillerons les concepts sous jacents aux bases de données relationnelles et, ensuite, nous expliquerons le mécanisme des modèles Entité-Association. Nous ne présenterons pas cette première partie en détail car celle-ci a fait l'objet d'un de nos cours aux facultés et ne constituait donc pas, pour nous, une matière nouvelle.

Dans un second lieu, nous aborderons le fonctionnement du prototype Anthémis selon le mécanisme des modèles Entité-Association. Cette partie sera, quant à elle, détaillée car elle représente avec le septième chapitre le sujet de notre stage à Montréal. Au sein de cette seconde partie, nous examinerons différents points reprenant la création du prototype dans son ensemble.

Nous détaillerons l'architecture de la base de données. Quelle structure devra suivre la base de données pour permettre la gestion des connaissances ? Nous examinerons aussi le mécanisme permettant l'introduction du modèle et des connaissances dans la base de données. Nous verrons, par après, le but des fichiers XSL au sein du prototype. Ces fichiers XSL comme vu dans le cinquième chapitre sont fortement liés aux fichiers XML. Et nous terminerons par les évolutions que nous avons entreprises entre les différentes versions du prototype anthémis.

I. Le formalisme des bases de données relationnelles

Avant de parler précisément du prototype Anthémis, nous allons faire un rappel des bases de données relationnelles et du modèle Entité-Association. Nous ne rentrerons pas dans les détails dans nos explications car ce formalisme et ce modèle nous ont été enseignés aux facultés.[9]

Les systèmes de gestion des bases de données (SGBD) doivent respecter plusieurs règles : garantir la qualité des données enregistrées, la cohérence de celles-ci, la restauration de celles-ci en cas d'incident, ainsi que de bonnes performances d'accès aux utilisateurs et permettre un accès parallèle aux données.

Il existe plusieurs types de SGBD différents qui représentent les données de façon différente comme le modèle hiérarchique, le modèle objet, etc. Mais le modèle relationnel est le plus répandu.

1. Concepts des bases de données relationnelles

Le modèle relationnel est basé sur une organisation des données sous forme de tables à deux dimensions (les lignes et les colonnes). Selon Pillou, « La manipulation des données se fait selon le concept mathématique de relation de la théorie des ensembles, c'est-à-dire l'algèbre relationnelle. L'algèbre relationnelle a été inventée en 1970 par Codd, le directeur de recherche du centre IBM de San José. Les opérations relationnelles permettent de créer une nouvelle relation (table) à partir d'opérations élémentaires sur d'autres tables ».[PILLOU01]

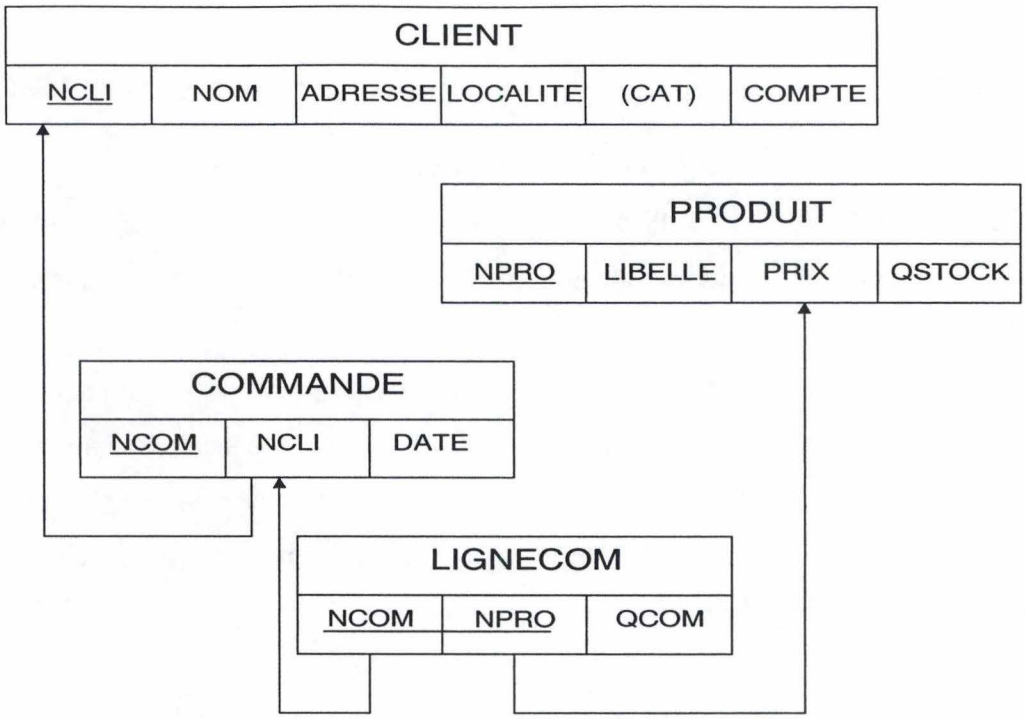
[9] Pour de plus amples informations, nous vous renvoyons au livre *'Bases de données et modèles de calcul. Outils et méthodes pour l'utilisateur.'* écrit par Jean-Luc Hainaut InterEditions 1994.

Selon le Professeur Hainaut, « une base de données relationnelle apparaît comme une collection de tables de données, ou de fichiers plats. Il s'agit d'une structure extrêmement simple et intuitive, qui pour l'utilisateur du moins, ne s'encombre d'aucun détail technique concernant les mécanismes de stockage sur disque et d'accès aux données. (...) Les données se présentent sous la forme de tables formées de lignes et de colonnes. Chaque ligne représente une entité ou un fait de la réalité, tandis qu'une colonne représente une propriété de ces entités ou faits. Une table contient donc des informations similaires sur une population d'entités ou de faits de la réalité. Certaines colonnes ont pour but d'identifier les lignes (identifiants), d'autres sont des références vers d'autres lignes (colonnes de référence et contraintes référentielles). ». [HAINAUT94, p12]

La structure d'une base de données relationnelle sera donc constituée de tables, de lignes et de colonnes. Nous pouvons définir ces notions de la manière suivante :

- Base de données : une base de données est une collection de tables, portant sur le même sujet.
- Tables : Toute information dans une base de données relationnelle se présente sous forme d'une table. Celle-ci est une collection de colonnes (sa structure) et de lignes (son contenu).
- Colonnes : Les colonnes représentent les propriétés des éléments contenus dans la table.
- Lignes : Les lignes représentent chacune des occurrences de la table.

La manière dont une base de données relationnelle se formalise est explicitée dans le SCHEMA 6.1. D'un point de vue graphique, les colonnes identifiantes sont soulignées par un trait, les colonnes facultatives sont mises entre parenthèses, les colonnes référentielles ont une flèche qui les relie à la table référencée.



SCHEMA 6.1 : Schéma d'une base de données relationnelle. [HAINAUT94, p26]

A chaque instant, la base de données doit vérifier qu'elle est dans un état cohérent. Pour cela il existe des contraintes d'intégrité exprimées par les colonnes identifiantes, les contraintes référentielles et les colonnes obligatoires ou facultatives. Si ces contraintes sont violées, nous dirons que les données ont perdu leur intégrité.

Tout ce qui concerne le travail sur la base de données, que ce soit de la consultation, du stockage ou de la mise à jour, s'effectue par le moyen du langage SQL (Structured Query Language). Le Professeur Hainaut nous apprend qu' « une instruction SQL constitue une requête, c'est-à-dire la description d'une opération que le SGBD doit exécuter. Une requête peut être introduite au terminal, auquel cas le résultat éventuel (dans le cadre d'une consultation de données par exemple) de l'exécution de la requête apparaît à l'écran. Cette requête peut également être envoyée par un programme au SGBD. Dans ce cas, le résultat de la requête est rangé par le SGBD, ligne par ligne, dans des variables du programme. ». [HAINAUT94, p36]

2. Modèle Entité-Association

Pour construire une base de données relationnelle, le moyen le plus simple est de créer ce qui est appelé 'un schéma conceptuel'. Ce dernier est souvent représenté par un graphique du modèle Entité-Association pour ensuite être traduit en structure de tables.

Le modèle Entité-Association permet de représenter naturellement les différents concepts que nous souhaitons formaliser dans la base de données sans s'attacher à tous les aspects techniques liés à l'introduction de ces concepts dans la base de données elle-même. Cette représentation se réalise sous forme graphique qui est simple à comprendre et attrayante pour l'utilisateur.

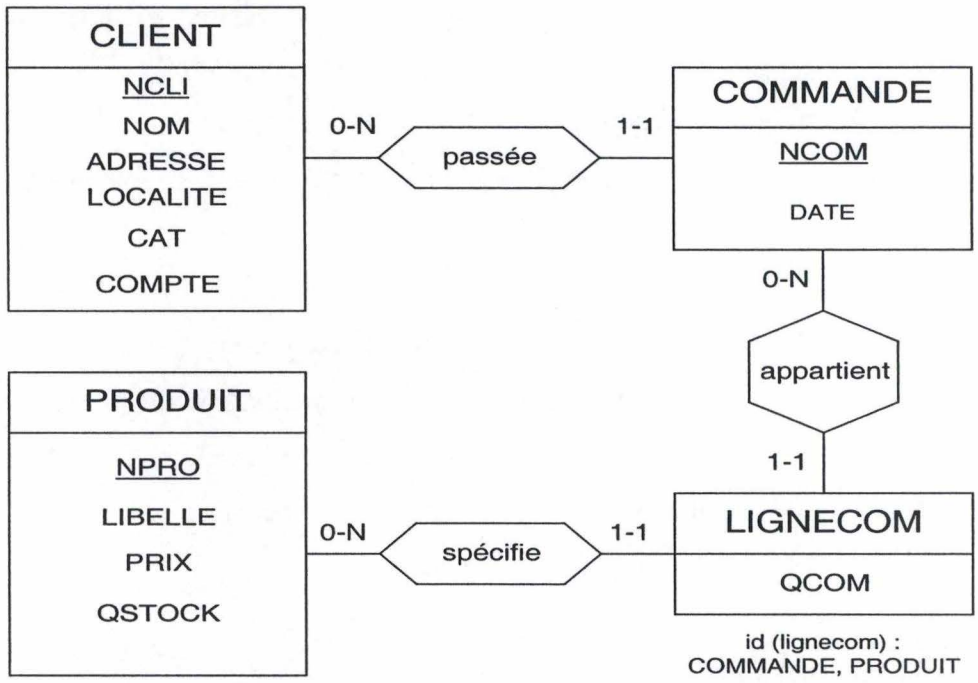
Le modèle Entité-Association est perçu selon le Professeur Hainaut « sous la forme d'ensembles d'entités. Les entités sont en association les unes avec les autres. Les entités d'une classe sont du même type. Elles sont caractérisées par des attributs qui décrivent leurs propriétés intrinsèques. Les types d'associations sont caractérisés par leur classe, qui indique combien d'entités d'un type sont associées à une entité de l'autre type, et par leur caractère obligatoire ou facultatif pour les entités associées. Un type d'entités a un identifiant, généralement constitué d'attributs. ». [HAINAUT94, p73]

Une entité est la représentation graphique d'une collection d'objets, concrets ou abstraits, sur lesquels on désire conserver des informations. Une entité est représentée graphiquement par un rectangle, avec son nom dans la partie supérieure, et les attributs dans la partie inférieure. On souligne le ou les attributs qui forme(nt) l'identifiant de cette entité.

Une relation est une association significative entre deux ou plusieurs entités. Une relation existe par l'entremise des entités qu'elle associe. Il est donc impossible d'adresser une occurrence d'une relation sans adresser les occurrences des entités qui y participent. Une relation est représentée graphiquement par un losange, avec son nom dans la partie supérieure, et les attributs dans la partie inférieure si elle en possède.

Alors que les entités et les relations déterminent la structure du modèle, et donc de la base de données, les attributs constituent réellement l'information qui sera conservée dans cette base de données. Les éléments de données constituent la plus petite parcelle d'information qui conserve encore un sens, une signification. Chaque attribut est obligatoirement inclus dans une entité ou une relation.

Nous allons maintenant reprendre les tables du SCHEMA 6.1 pour les modéliser dans le formalisme du modèle Entité-Association. Nous verrons que ce schéma est intuitif et qu'il permet une bonne compréhension dès la première lecture. (SCHEMA 6.2)



SCHEMA 6.2 : Schéma conceptuel correspondant au schéma 6.1. [HAINAUT94, p83]

II. Prototype de gestion de la connaissance par le modèle Entité-Association

Nous avons donc vu dans la première partie de ce chapitre que les bases de données relationnelles étaient les plus répandues à l'heure actuelle. Notre prototype Anthémis utilise le formalisme des bases de données relationnelles. Pour gérer cette base de données, nous avons utilisé le SGBD ORACLE vu dans le quatrième chapitre du présent travail.

Cette seconde partie est consacrée à la création du prototype Anthémis en se basant sur le principe des modèles Entité-Association dont nous venons de donner une rapide explication. Un des pré-requis demandé à l'utilisateur du prototype est d'être capable de modéliser ses connaissances dans un modèle Entité-Association. Ce pré-requis ne nous semble pas trop lourd vu la facilité avec laquelle on rentre dans le raisonnement du modèle Entité-Association.

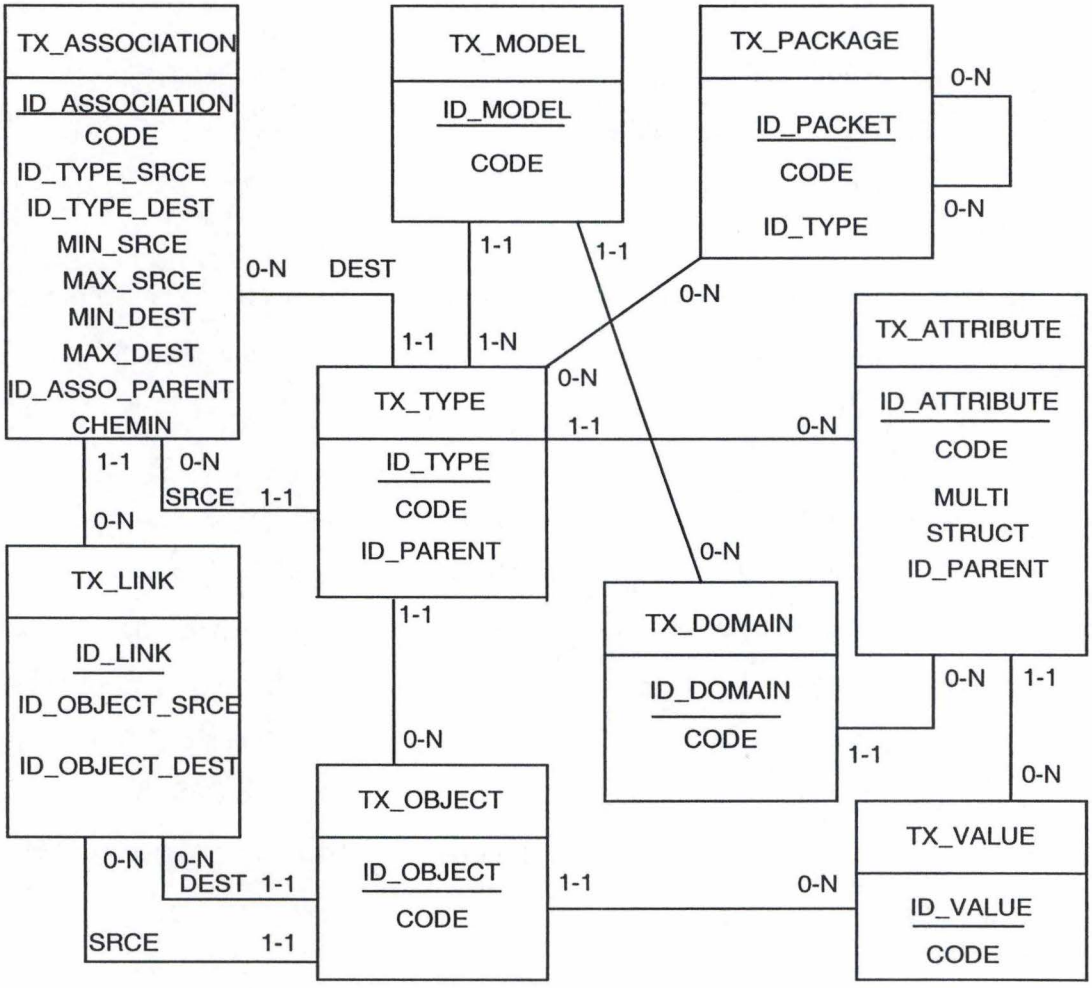
Nous allons maintenant détailler le fonctionnement du prototype via le modèle Entité-Association. Pour cela, nous expliquerons d'abord l'architecture de la base de données. Ensuite, nous aborderons le mécanisme d'introduction des connaissances dans la base de données. Nous verrons enfin le traitement de ces connaissances.

1. Architecture de la base de données

La conception de la structure de la base de données a été réalisée par Mademoiselle Morin. Avant notre arrivée, cette dernière travaillait sur le prototype Anthémis dans le cadre d'un stage semblable au nôtre. Nous avons gardé sa structure tout en y apportant quelques modifications. Nous verrons tout de même, par la suite, que cette structure est susceptible d'être améliorée.

Le cas des mémoires d'entreprises nous conduisent à devoir gérer tous les types de connaissances, que ce soit les connaissances explicites ou tacites dont nous avons vu les définitions dans le premier chapitre. Il faut dès lors avoir une structure suffisamment flexible pour gérer ces différentes connaissances. Nous expliquerons ce terme de flexibilité

de la base de données par après. Nous allons donc représenter dans le SCHEMA 6.3 le modèle que suivra la base de données (nous ne placerons pas dans ce modèle les associations par souci de clarté du schéma).



SCHEMA 6.3 : Structure de la base de données pour le modèle Entité-Association.

La structure de la base de données contient deux niveaux : le niveau modèle et le niveau connaissance. Le niveau modèle permet la construction d'un modèle de données. Un modèle est ainsi une façon de structurer les données. Le niveau connaissance, quant à lui, permet d'entrer des connaissances dans le modèle. C'est grâce à ces deux niveaux que nous pouvons effectivement avoir une base de données flexible. En effet toutes les connaissances introduites dans la base de données qui sont les données de cette base devront suivre un modèle. Or ce modèle est lui même introduit dans la base de données comme une donnée. Nous ne sommes donc pas limités à introduire des connaissances

suivant un unique modèle mais bien suivant une multitude de modèles définis par l'utilisateur lui même.

1.1. Le niveau modèle

Nous retrouvons dans le niveau modèle la table TX_MODEL, la table TX_TYPE, la table TX_ASSOCIATION, la table TX_ATTRIBUTE, la table TX_DOMAIN ainsi que la table TX_PACKAGE.

La table TX_MODEL permet de définir le modèle sur lequel nous sommes en train de travailler. Elle permet d'avoir plusieurs modèles différents dans la base de données.

La table TX_TYPE permet de rassembler l'ensemble des types du modèle. Un type est une catégorie regroupant un ensemble d'objets ayant des propriétés similaires. Les types sont caractérisés par leurs attributs et leurs associations. Dans tous les modèles, nous retrouvons le **type universel**. Le type universel reprend l'ensemble des types qui n'ont pas de parents spécifiés par l'utilisateur. En d'autres mots, ce sont les types qui n'ont pas de sur-types. Les types universels prennent la valeur '0' pour ID_PARENT. Il faut savoir qu'un type qui a un parent, donc un sur-type, reprend les attributs de ce sur-type. C'est le principe de l'héritage.

La table TX_ASSOCIATION reprend l'ensemble des associations du modèle. Une association est une liaison entre deux types. Nous pouvons avoir des associations simples ainsi que des associations virtuelles. Les associations simples sont des liaisons entre deux types qui sont des voisins proches alors que les associations virtuelles lient des types qui sont des voisins éloignés. Pour se rendre chez un voisin éloigné, nous devons suivre un chemin et ce chemin représente une série d'associations simples.

La table TX_ATTRIBUTE rassemble l'ensemble des attributs. Un attribut définit une caractéristique d'un type. Il représente donc une information commune à l'ensemble des objets de ce type. Un type peut avoir plusieurs attributs qui lui sont propres et il peut hériter de ceux de ses parents. Un attribut peut être simple, multivalué ou structuré. Un

hériter de ceux de ses parents. Un attribut peut être simple, multivalué ou structuré. Un attribut peut donc avoir des sous-attributs s'il est multivalué ou structuré. Pour un sous-attribut, nous devons passer l'identifiant du parent. Si un attribut ne possède pas de parents, il aura la valeur '0' pour ID_PARENT.

La table TX_DOMAIN reprend les domaines du modèle. Un attribut possède un domaine qui nous permet de savoir quel genre d'information il peut contenir (entier, caractère, booléen, etc.).

La table TX_PACKAGE rassemble l'ensemble des paquets du modèle. Un paquet est un regroupement de types qui proviennent de modèle existant. Un paquet sert principalement à la réutilisation des types. Ainsi un type existant dans un modèle peut être réutilisé dans un autre modèle.

1.2. Le niveau connaissance

Le niveau connaissance contient la table TX_OBJECT, la table TX_LINK et la table TX_VALUE.

La table TX_OBJECT regroupe l'ensemble des objets. Les objets sont des instances des types qui représentent des entités uniques du monde réel. Si nous voulons créer un objet 'Jean-Philippe', nous devons spécifier qu'il s'agit d'une personne et que Jean-Philippe est un objet du type 'personne'.

La table TX_LINK rassemble l'ensemble des liens. Les liens sont des instances des associations. Lors de la création d'un lien entre deux objets, nous vérifions que les deux types d'où proviennent ces objets ont bien l'association correspondante entre eux et que les cardinalités de l'association sont bien respectées.

La table TX_VALUE reprend l'ensemble des valeurs. Les valeurs sont des instances des attributs. Si nous avons le type 'personne', si ce type a comme attributs 'tel' et 'nom' et si nous avons l'objet 'Jean-Philippe' représentant une personne, alors cet objet aura comme valeurs '071/43.80.55' et 'Rochet'.

Pour le détail de la création de cette base de données, nous reportons au fichier initBD.asp de l'ANNEXE 2, permettant d'initialiser la base de données. L'ANNEXE 2 reprend le code de tous les fichiers du prototype.

2. Introduction du modèle et des connaissances

Passons maintenant à l'introduction des connaissances dans le prototype. Pour introduire les connaissances dans le prototype, l'utilisateur doit agir en deux temps. Tout d'abord, il doit introduire le modèle (la structure) que vont suivre ses connaissances. Ensuite, il devra introduire ses propres connaissances qui dès lors suivront le modèle précédemment introduit. L'introduction des modèles et des connaissances se réalise au moyen de fichiers XML.

2.1. Introduction du modèle

L'utilisateur doit dans un premier temps concevoir le modèle que ses connaissances devront suivre. Pour concevoir ce modèle, un pré-requis nécessaire pour l'utilisateur se retrouve dans la modélisation de ses idées sous forme de modèle Entité-Association. Une fois ce modèle créé, il reste à l'utilisateur à traduire ce modèle dans un formalisme XML pré-écrit par nos soins. Pour cette traduction, il faut que chaque utilisateur du prototype connaisse les différentes balises que nous avons établies et grâce auxquelles l'utilisateur pourra introduire son modèle. Nous allons étudier ces balises et leur agencement dans les sections suivantes. Il faut remarquer que cette traduction du modèle Entité-Association vers un fichier XML devra à long terme disparaître. Nous avons vu dans le troisième chapitre une possibilité de traduction par le programme DB-Main.

2.1.1. Formalisme XML

Nous allons maintenant examiner les balises XML établies qui permettront l'écriture du modèle de l'utilisateur. Ces balises ont été créées de manière à garantir à

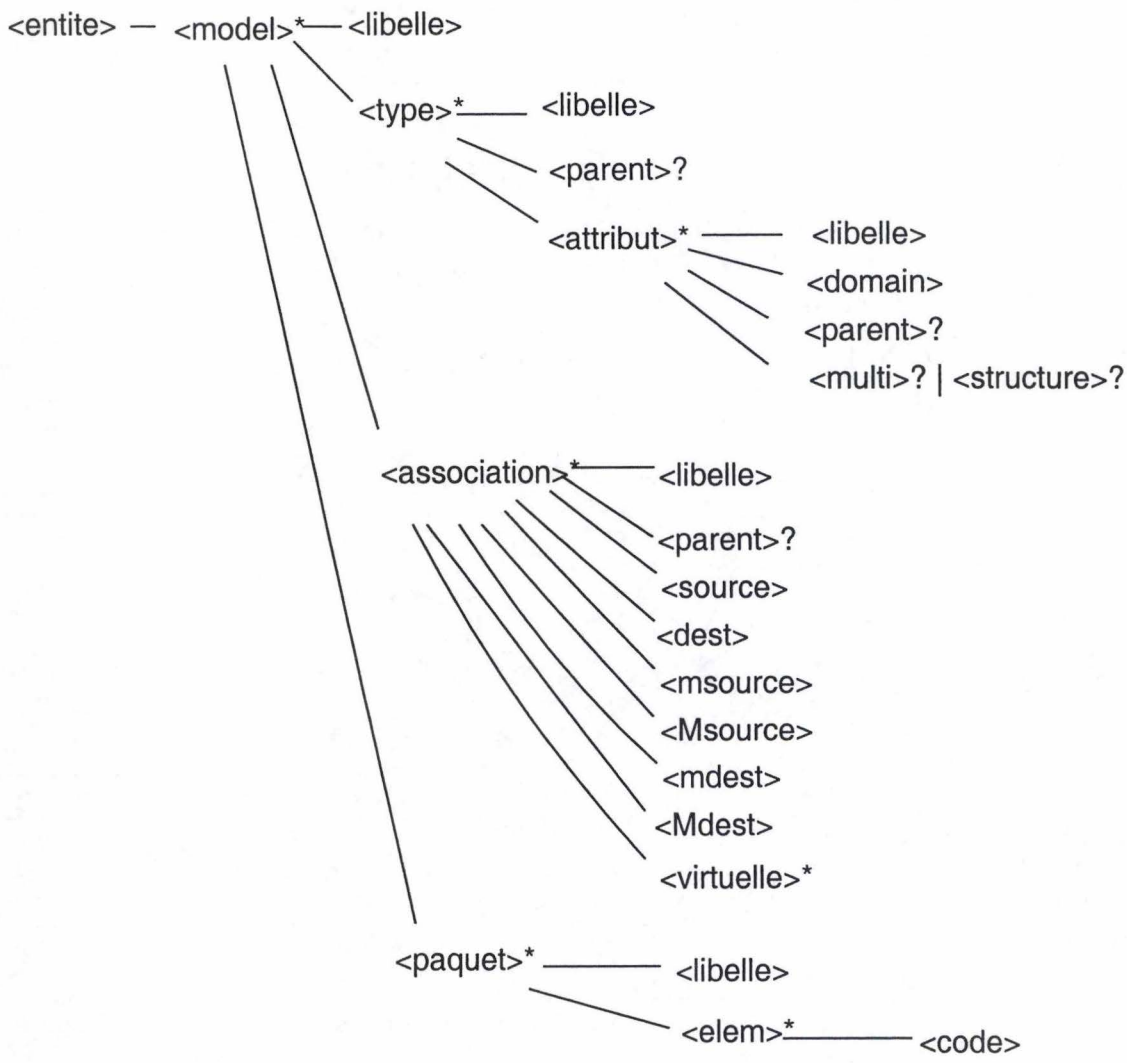
l'utilisateur une utilisation aisée. Elles sont évidemment toutes accompagnées d'une balise fermante. Nous reprenons les différentes balises dans le TABLEAU 6.1.

BALISES	DESCRIPTION
<entite>	Cette balise est la balise racine, elle signale que nous sommes en train d'écrire un fichier XML qui décrit un modèle.
<model>	Cette balise permet de définir un nouveau modèle.
<type>	Cette balise offre la possibilité de définir un nouveau type.
<association>	Cette balise permet de définir une nouvelle association.
<attribut>	Cette balise propose de définir un nouvel attribut.
<paquet>	Cette balise offre la possibilité de gérer des paquets.
<libelle>	Cette balise est toujours contenue au sein d'une autre balise. Elle donne le libellé de la balise qui la contient.
<parent>	Cette balise est, elle aussi, toujours contenue dans une autre balise. Elle donne le parent de la balise qui la contient.
<domain>	Cette balise permet de définir le domaine d'un attribut.
<multi> ou <structure>	Ces balises permettent d'établir si un attribut est multivalué ou structuré.
<source>, <dest>	Ces balises permettent pour une association de définir le type qui sera la source et le type qui sera la destination de l'association.
<msource>, <Msource>	Ces balises permettent de définir la cardinalité minimale et Maximale pour le type source.
<mdest>, <Mdest>	Ces balises permettent de définir la cardinalité minimale et Maximale pour le type destination.
<virtuelle>	Cette balise permet de définir le chemin pour une association virtuelle.
<elem>	Cette balise permet d'énumérer les types repris dans le paquet.
<code>	Cette balise permet d'énoncer les types repris dans la balise <elem> d'un paquet.

TABLEAU 6.1 : Balises XML utilisées pour introduire un modèle.

2.1.2. Agencement des balises

Nous allons, maintenant, représenter sous la forme d'un arbre l'agencement des différentes balises du modèle. Un fichier XML reprenant le modèle que l'on veut formaliser devra toujours présenter cet agencement. (SCHEMA 6.4)



SCHEMA 6.4 : Agencement des balises concernant le modèle.

2.1.3. Création d'une DTD pour le modèle

Nous allons à présent écrire la DTD qui nous signalera quels fichiers XML seront reconnus valides lors de l'insertion des fichiers XML dans le prototype. (SCHEMA 6.5)

```
< !ELEMENT entite (model+)>
< !ELEMENT model (libelle, type*, association,*, paquet*)>
< !ELEMENT libelle (#PCDATA)>
< !ELEMENT type (libelle, parent ?, attribut*)>
< !ELEMENT libelle (#PCDATA)>
< !ELEMENT parent (#PCDATA)>
< !ELEMENT attribut (libelle, domain, multi ? | structure ?, parent ?)>
< !ELEMENT libelle (#PCDATA)>
< !ELEMENT domain (#PCDATA)>
< !ELEMENT multi (#PCDATA)>
< !ELEMENT structure (#PCDATA)>
< !ELEMENT parent (#PCDATA)>
< !ELEMENT association (libelle, parent ?, source, dest, msource, Msource, mdest,
Mdest, virtuelle*, attribut*)>
< !ELEMENT libelle (#PCDATA)>
< !ELEMENT parent (#PCDATA)>
< !ELEMENT source (#PCDATA)>
< !ELEMENT dest (#PCDATA)>
< !ELEMENT msource (#PCDATA)>
< !ELEMENT Msource (#PCDATA)>
< !ELEMENT mdest (#PCDATA)>
< !ELEMENT Mdest (#PCDATA)>
< !ELEMENT virtuelle (#PCDATA)>
< !ELEMENT attribut (libelle, domain, multi ? | structure ?, parent ?)>
< !ELEMENT libelle (#PCDATA)>
< !ELEMENT domain (#PCDATA)>
< !ELEMENT multi (#PCDATA)>
< !ELEMENT structure (#PCDATA)>
< !ELEMENT parent (#PCDATA)>
< !ELEMENT paquet (libelle, elem*)>
< !ELEMENT libelle (#PCDATA)>
< !ELEMENT elem (code)>
< !ELEMENT code (#PCDATA)>
```

SCHEMA 6.5 : DTD pour les fichiers XML concernant le modèle.

2.2. Introduction des connaissances

L'introduction des connaissances suit le même principe que celui du modèle mais est légèrement plus compliqué car il doit faire référence à la structure du fichier XML précédemment créé concernant le modèle.

2.2.1. Formalisme XML

Nous allons examiner les balises utilisées pour pouvoir insérer les connaissances de l'utilisateur. Ces balises devront suivre une structure permettant de faire référence aux éléments intégrés dans le fichier XML concernant le modèle. Il nous est ici impossible de faire une énumération précise des balises utilisées car celles-ci sont créées en rapport avec le fichier XML du modèle. Il nous est donc, également, impossible de créer une DTD concernant le fichier XML des connaissances.

Souvenons nous, nous avons vu que dans le niveau connaissance, nous retrouvions trois tables : la table TX_OBJECT, la table TX_LINK et la table TX_VALUE. Les balises devront donc être créées autour de ces trois tables. Nous savons aussi que le niveau connaissance représente des instances du niveau modèle. La table TX_OBJECT en correspondance avec la table TX_TYPE, la table TX_LINK avec la table TX_ASSOCIATION et la table TX_VALUE avec la table TX_ATTRIBUTE.

Nous dénommerons la référence vers un élément du fichier XML du modèle par le terme **référent**, ainsi que la valeur de la table par **valeur**. Nous nous trouverons donc devant des balises de la forme suivante : < **référent** table = '...'> **valeur** </**référent**>. Nous allons reprendre dans le TABLEAU 6.2 la forme des différentes balises. Plus généralement ces balises fonctionnent avec des attributs qui sont **table** et **table_TARGET**.

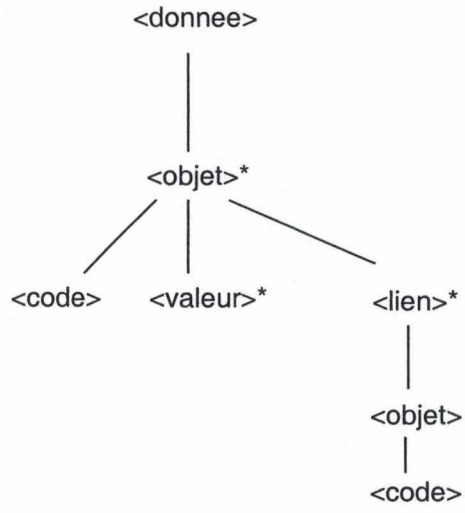
BALISES	DESCRIPTION
<donnee>	Cette balise est la balise racine, elle signale que nous sommes en train d'écrire un fichier XML qui décrit des connaissances.
<**référent** table = 'object'> (...) </**référent**>	Ce type de balise permet d'insérer un nouvel objet. Nous retrouvons à la place des (...) d'autres balises qui sont <code> et celles concernant les valeurs et les liens. A la place de **référent**, nous aurons un type existant dans le fichier XML du modèle.
<code>	Cette balise permet, uniquement pour un objet, d'insérer le libellé de l'objet.
<**référent** table = 'value'> **valeur** </**référent**>	Ce type de balise permet d'insérer une nouvelle valeur. A la place de **référent**, nous aurons un attribut existant dans le fichier XML du modèle.
<**référent** table = 'value' />	Ce type de balise est utilisée pour des attributs multivalués ou structurés. A la place de **référent**, nous aurons un attribut multivalué ou structuré existant dans le modèle.

<**référent** table = 'link' table_TARGET = '[...]'> (...) </**référent**>	Ce type de balise permet d'insérer un nouveau lien. Elle se trouve toujours au sein d'une balise 'object'. Objet qui sera soit la source ou la destination du lien. Nous retrouvons à la place des (...) l'objet qui sera l'autre bout du lien. Si le mot clef entre [...] est srce (dest) alors l'objet entre (...) est la source (la destination). A la place de **référent**, nous aurons une association existante dans le fichier XML du modèle.

TABLEAU 6.2 : Balises XML utilisées pour introduire des connaissances.

2.2.2. Agencement des balises

Nous allons représenter l'agencement des différentes balises. Par souci de clarté, nous dénommerons les balises concernant les objets par <objet>, celles concernant les valeurs par <valeur> et celles concernant les liens par <lien>. (SCHEMA 6.6)



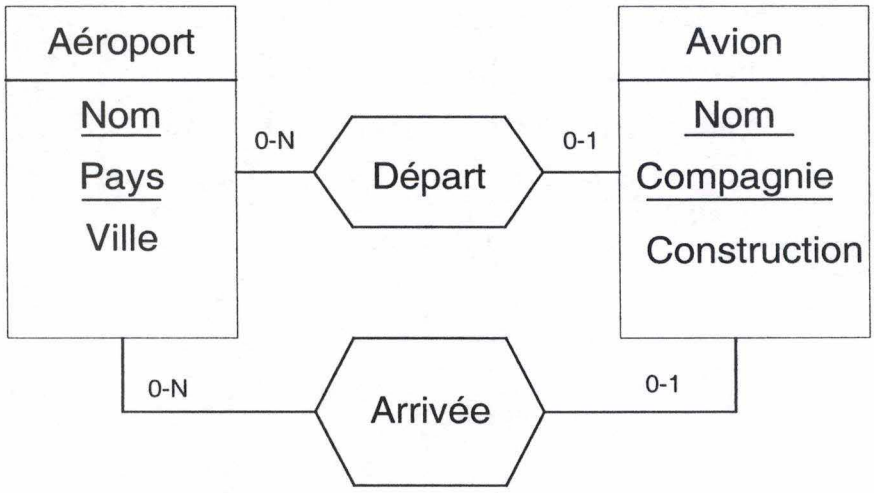
SCHEMA 6.6 : Agencement des balises concernant les connaissances.

2.2.3. Etude de cas

Nous allons réaliser un petit cas explicatif dans lequel nous simulerons ce que l'utilisateur devra réaliser comme manipulations pour pouvoir introduire ses connaissances dans le prototype.

Premièrement, l'utilisateur devra énoncer en langage naturel les connaissances qu'il veut représenter. Nous prendrons, ici, un cas assez simple de gestion d'aéroport : L'avion BX2000 d'Air Canada construit en 1988 part de l'aéroport de Zaventem de Bruxelles en Belgique et arrive à l'aéroport de Dorval de Montréal au Québec.

Deuxièmement, l'utilisateur pourra, par simplicité, réaliser sur papier le modèle Entité-Association représentant la connaissance qu'il va vouloir intégrer au système. (SCHEMA 6.7) Ce modèle dans le cadre d'une réelle gestion d'aéroport devrait être beaucoup plus complet. Nous ne sommes pas rentrés dans les détails car ce cas sert à expliquer l'utilisation des fichiers XML et non à montrer les capacités de cette technique.



SCHEMA 6.7 : Représentation du modèle Entité-Association pour le cas.

Troisièmement, l'utilisateur devra traduire ce modèle en fichier XML suivant les balises prédéfinies et citées ci-dessus.

```
<entite>
    <model>
        <libelle> Gestion d'aéroport </libelle>
        <type>
            <libelle> Aéroport </libelle>
            <attribut>
                <libelle> Nom </libelle>
                <domain> string </domain>
            </attribut>
            <attribut>
                <libelle> Pays </libelle>
                <domain> string </domain>
            </attribut>
            <attribut>
                <libelle> Ville </libelle>
                <domain> string </domain>
            </attribut>
        </type>
        <type>
            <libelle> Avion </libelle>
            <attribut>
                <libelle> Nom </libelle>
                <domain> string </domain>
            </attribut>
            <attribut>
                <libelle> Compagnie </libelle>
                <domain> string </domain>
            </attribut>
            <attribut>
                <libelle> Construction </libelle>
                <domain> entier </domain>
            </attribut>
        </type>
        <association>
            <libelle> Départ </libelle>
            <source> Avion </source>
            <dest> Aéroport </dest>
            <msource> 0 </msource>
            <Msource> 1 </Msource>
            <mdest> 0 </mdest>
            <Mdest> N </Mdest>
        </association>
        <association>
            <libelle> Arrivée </libelle>
            <source> Avion </source>
            <dest> Aéroport </dest>
            <msource> 0 </msource>
```

```
                    <Msource> 1 </Msource>
                    <mdest> 0 </mdest>
                    <Mdest> N </Mdest>
                </association>
        </model>
</entite>
```

Quatrièmement, l'utilisateur devra introduire dans son modèle les connaissances désirées. Cette opération se réalise, elle aussi, par l'intermédiaire d'un fichier XML.

```
<donnee>

        <Aéroport table = 'object'>
                <code> Aéroport1 </code>
                <Nom table = 'value'> Zaventem </Nom>
                <Pays table = 'value'>  Belgique </Pays>
                <Ville table = 'value'> Bruxelles </Ville>
        </Aéroport>
        <Aéroport table = 'object'>
                <code> Aéroport2 </code>
                <Nom table = 'value'> Dorval </Nom>
                <Pays table = 'value'> Québec </Pays>
                <Ville table = 'value'> Montréal </Ville>
        </Aéroport>
        <Avion table = 'object'>
                <code> Avion1 </code>
                <Nom table = 'value'> BX2000 </Nom>
                <Compagnie table = 'value'> Air Canada </Compagnie>
                <Construction table = 'value'> 1986 </Construction>
                <Départ table = 'link' table_TARGET = 'dest'>
                        <Aéroport table = 'object'>
                                <code> Aéroport1 </code>
                        </Aéroport>
                </Départ>
                <Arrivée table = 'link' table_TARGET = 'dest'>
                        <Aéroport table = 'object'>
                                <code> Aéroport2 </code>
                        </Aéroport>
                </Arrivée>
        </Avion>

</donnee>
```

D'autres exemples de fichiers XML de modèle ou de connaissance plus complexes se trouvent à l'ANNEXE 3 du présent travail.

3. Intervention des fichiers XSL

Une fois les fichiers XML pour le modèle et pour les connaissances créés, l'utilisateur va devoir intégrer ses connaissances dans la base de données. Pour ce faire, l'utilisateur devra charger ses fichiers XML dans le prototype. Ce chargement se réalise par l'interface graphique. Effectivement, dès le lancement du prototype, l'utilisateur se retrouve devant une page l'invitant à charger son modèle et ses connaissances. Cette fenêtre se trouve au troisième chapitre de ce présent travail. (ECRAN 3.1)

3.1. Fichier XSL pour le modèle

L'utilisateur doit donc passer le nom de son fichier XML sans l'extension *.xml de celui-ci. Une fois que l'utilisateur aura appuyé sur le bouton **charger** de la page **chargement.html**, le fichier **chargement.asp** réalisera le traitement de ce fichier. Ce fichier vérifiera qu'il a bien à faire à un fichier concernant les modèles et appliquera dès lors le fichier **creermodel.xsl**.

Le fichier **chargement.asp** s'occupera également de la fusion entre le fichier XML concernant les modèles et le fichier XSL concernant les modèles grâce à la ligne de code **var result = source.transformNode(style);** avec source contenant le fichier XML et style contenant le fichier XSL. Il créera alors un nouveau fichier qui prendra le même nom que le fichier XML et qui aura l'extension *.txs. Ce fichier sera utilisé par le fichier **creermodel.asp** pour le stockage dans la base de données.

Ainsi, dans le cas où un utilisateur crée le fichier XML '**monmodele.xml**', il doit alors écrire dans la fenêtre d'ouverture du prototype '**monmodele**'. Une fois cela fait, il doit appuyer sur le bouton **charger**. C'est alors le prototype qui s'occupe de tout et qui

introduit le modèle dans la base de données en ayant créé entre temps un fichier temporaire **monmodele.txs**.

Pour une vision plus approfondie, nous recommandons de lire, dans l'ordre, le code des fichiers chargement.html, chargement.asp, creermodel.xsl, un exemple de fichier *.txs ainsi que le fichier creermodel.asp. Tous ces fichiers se trouvent à l'ANNEXE 2 du présent travail sauf l'exemple de fichier *.txs qui se trouve à l'ANNEXE 3.

3.2. Fichier XSL pour les connaissances

L'utilisateur, comme pour le modèle, doit passer le nom de son fichier XML sans l'extension *.xml de celui-ci et appuyer sur le bouton **charger** de la page **chargement.html**. Le fichier **chargement.asp** réalisera le traitement de ce fichier. Ce fichier vérifiera qu'il se trouve devant un fichier concernant les connaissances et qu'un modèle a été au préalable inséré. Il appliquera dès lors le fichier **creerdonnee.xsl**.

Comme pour le modèle, le fichier **chargement.asp** s'occupera de la fusion entre le fichier XML concernant les connaissances et le fichier XSL concernant les connaissances. Il créera également un nouveau fichier qui prendra le même nom que le fichier XML et qui aura l'extension *.txs. Ce fichier sera utilisé par le fichier **creerdonnee.asp** pour le stockage dans la base de données.

Pour une vision plus approfondie, nous recommandons de lire, dans l'ordre, le code des fichiers chargement.html, chargement.asp, creerdonnee.xsl, un exemple de fichier *.txs ainsi que le fichier creerdonnee.asp. Tous ces fichiers se trouvent à l'ANNEXE 2 du présent travail sauf l'exemple de fichier *.txs qui se trouve à l'ANNEXE 3.

4. Evolutions du prototype Anthémis

Avant de parler des évolutions entreprises, il faut savoir qu'il existe trois versions du prototype. La première est celle présentée dans ce travail. Elle est la plus stable et

fonctionne dans sa totalité (à l'exception de la présentation). La deuxième version comprend les attributs des associations et repose sur une structure de base de données différente de celle présentée dans ce travail (SCHEMA 6.3). Elle fonctionne également (sauf pour la présentation). La troisième version, quant à elle, ne fonctionne pas et n'a pas été terminée durant le stage.

Pourquoi avoir réalisé plusieurs versions du prototype ? Simplement car le Professeur Gerbé, une fois la première version du prototype terminée, nous a demandé de retravailler sur la base de données en essayant d'obtenir une structure de tables se rapprochant des graphes conceptuels. Une solution possible pour diminuer le nombre de tables de la base de données était de rassembler les entités concernant le modèle avec les entités concernant les connaissances, ce qui a donné la deuxième version. La troisième version, quant à elle, provient du fait que lorsque le Professeur Gerbé a examiné le code de la seconde version, il s'est rendu compte que notre code n'était pas en accord avec son code concernant les fichiers de présentation. Il a donc fallu retravailler une fois de plus la structure de la base de données en tenant compte cette fois du code de présentation.

Le prototype, lorsque nous avons stoppé nos travaux, se composait de deux versions. La première version, qui fera l'objet du sixième chapitre du présent travail, reprend une base de données composée de neuf tables. L'élément important de cette base de données est la division de celle-ci en deux niveaux : le niveau modèle, qui gère les modèles insérés par l'utilisateur et le niveau connaissance, qui gère les connaissances liées au modèle de l'utilisateur. La seconde version, quant à elle, reprend une base de données composée de six tables et permet la gestion des attributs pour les associations.

Pratiquement, les évolutions entreprises ont été tout d'abord la possibilité d'introduire des attributs aux associations, ensuite, la réduction du nombre de tables de la base de données.

Ainsi, la première évolution qui a été réalisée était de permettre aux associations d'avoir des attributs. Cette évolution a été entreprise et concrétisée sur le prototype mais uniquement dans la deuxième version. Pour permettre cette évolution, il a simplement suffit de modifier les fichiers **creermodel.xsl**, **creerdonnee.xsl** ainsi que le fichier **creerdonnee.asp**. Nous montrons ces modifications dans l'ANNEXE 4 du présent travail.

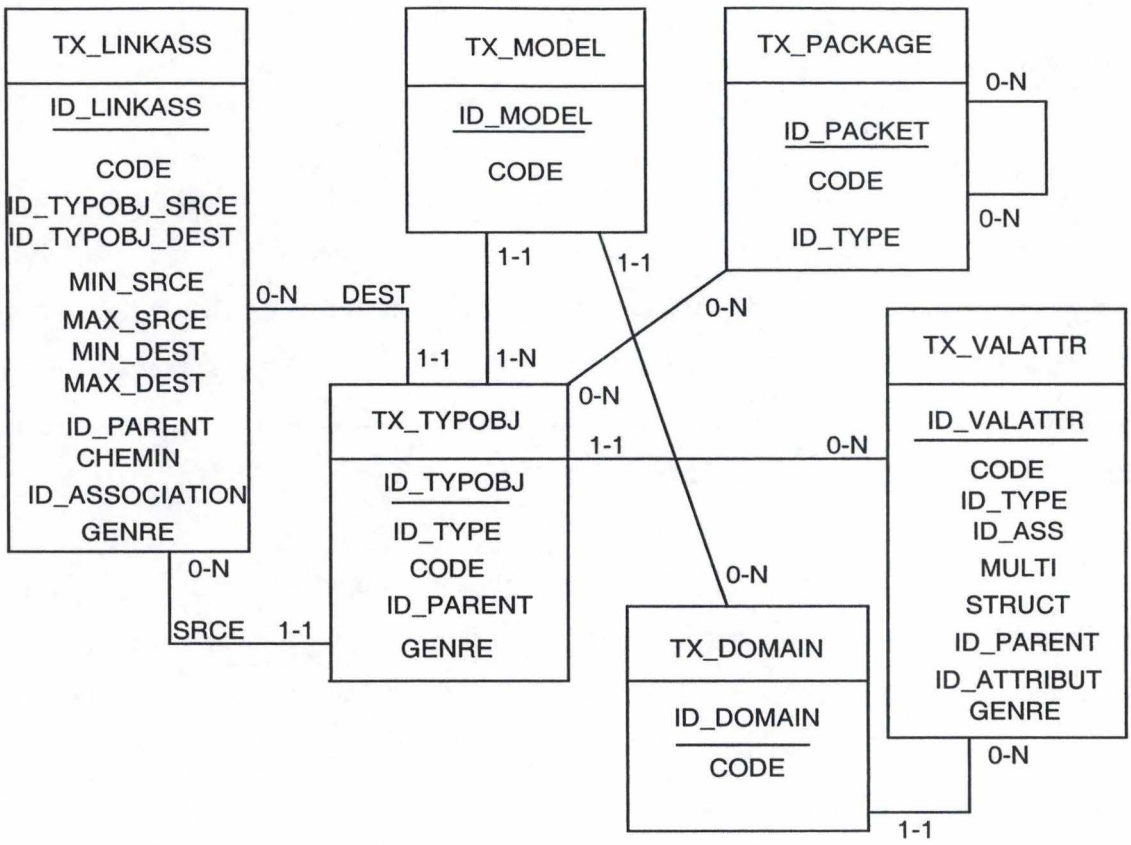
Si l'utilisateur veut introduire un attribut pour une association, celui-ci doit alors, dans son fichier XML concernant les connaissances, écrire la ligne de code reprise dans le TABLEAU 6.3. Ce nouveau tableau doit être considéré comme une nouvelle ligne du TABLEAU 6.2.

<**référent** table = 'avalue'> **valeur** </**référent**>	Ce type de balise permet d'insérer une nouvelle valeur. A la place de **référent**, nous aurons un attribut d'une association existant dans le fichier XML du modèle.

TABLEAU 6.3 : Balise XML supplémentaire pour introduire des connaissances.

La seconde évolution devait diminuer le nombre de tables de la base de données. Pour cela, il a fallu jumeler la table TX_TYPE avec la table TX_OBJECT donnant la table TX_TYPOBJ, la table TX_ASSOCIATION avec la table TX_LIEN donnant la table TX_LINKASS et la table TX_ATTRIBUT avec la table TX_VALUE donnant la table TX_VALATTR. Ce jumelage a eu comme résultat le passage de neuf tables à six tables. Il a fallu ajouter un attribut supplémentaire 'genre' à ces tables pour permettre de maintenir la distinction entre le niveau modèle et le niveau connaissance.

La structure de la base de données est alors devenue celle indiquée dans le SCHEMA 6.8.



SCHEMA 6.8 : Structure de la base de données pour la deuxième version.

Cependant, la manière dont nous avions jumelé les différentes tables ne correspondait pas à l'idée que le Professeur Gerbé s'était faite de son côté. Dès lors, Il a fallu reprendre une nouvelle fois, l'architecture de la base de données pour correspondre à cette idée. Cela nous a conduit à une troisième version du prototype qui ne faisait que commencer lors de notre départ du projet et qui a sans doute continué avec l'étudiante nous succédant.

Pour une autre compréhension de la structure de la base de donnée concernant la deuxième version du prototype, nous vous renvoyons au code du fichier **initBD.asp** se trouvant à l'ANNEXE 4.

Conclusion de chapitre

Nous avons, tout d'abord, dans ce chapitre parlé des bases de données relationnelles. Nous avons vu que celles-ci étaient les plus répandues à l'heure actuelle et qu'elles se formalisaient sous forme de tables. A chaque instant, la base de données vérifie qu'elle est dans un état cohérent grâce aux contraintes d'intégrité. Toutes les manipulations effectuées sur la base de données, que ce soit de la consultation, du stockage ou de la mise à jour, s'effectue par le moyen du langage SQL.

Nous avons ensuite expliqué le fonctionnement du modèle Entité-Association. Ce modèle permet de représenter naturellement les différents concepts que nous souhaitons formaliser dans la base de données sans s'attacher à tous les aspects techniques liés à l'introduction de ces concepts dans la base de données elle-même. Cette représentation se réalise sous forme graphique.

Par après, nous avons décrit le fonctionnement du prototype Anthémis selon les modèles Entité-Association. Pour cela, nous avons détaillé l'architecture de la base de données qui étaient composée de neuf tables. La base de données est divisée en deux niveaux : le niveau modèle et le niveau connaissance.

Nous avons détaillé l'introduction des connaissances dans le prototype. Pour introduire les connaissances dans le prototype, l'utilisateur doit agir en deux temps. Tout d'abord, il doit introduire le modèle (la structure) que vont suivre ses connaissances. Ensuite, il devra introduire ses propres connaissances qui dès lors suivront le modèle précédemment introduit. L'introduction des modèles et des connaissances se réalisent au moyen de fichiers XML. Nous avons également présenté un cas simple permettant d'illustrer le fonctionnement des fichiers XML.

Nous avons expliqué par la suite l'intervention des fichiers XSL qui servent à transformer les fichiers XML introduits par l'utilisateur en fichier TXS qui permettent l'intégration des modèles et des connaissances dans la base de données. L'explication des différents fichiers qui entrent en jeux est également décrite.

Vient finalement la dernière partie concernant les évolutions du prototype. Dans cette partie nous avons présenté des évolutions réalisées sur le prototype entre la première version présentée dans ce présent travail et la seconde version dont nous avons donné une brève explication.

Nous pouvons donc dire que le prototype selon le modèle Entité-Association répond à nos attentes, c'est-à-dire la création d'un logiciel permettant de gérer les connaissances. En effet, la base de données ainsi construite permet à l'utilisateur d'insérer n'importe quel modèle de connaissances. La présentation des connaissances suivra un modèle, donc les pages seront uniformes et standardisées. Reste le problème des pré-requis d'introduction des connaissances qui devront disparaître avec l'avancement du prototype.

Dans le chapitre suivant, nous allons parler du formalisme des graphes conceptuels et du fonctionnement du prototype correspondant à ce formalisme.

Septième chapitre

Le formalisme des graphes conceptuels pour gérer la connaissance

---

I. Introduction aux graphes conceptuels

1. Notion de concept et de relation
2. Hiérarchie des types
3. Forme logique et linéaire
4. Opérations de dérivation
5. Graphes conceptuels canons et généralisation
   - 5.1. Graphes conceptuels canons
   - 5.2. Généralisation

II. Le choix d'un formalisme

1. Introduction
2. Les besoins fondamentaux
3. Les formalismes étudiés
   - 3.1. Modèle Entité-Association
   - 3.2. UML
   - 3.3. Graphes conceptuels

III. Un métamodèle pour la représentation des connaissances

1. Introduction
2. Le métamodèle
   - 2.1. Les graphes
   - 2.2. Les éléments
   - 2.3. Les concepts
   - 2.4. Les référents
   - 2.5. Notation

IV. Anthémis, prototype de gestion de la connaissance par les graphes conceptuels

1. Architecture de la base de données
2. Insertion et notions de base
   - 2.1. Concepts de base
   - 2.2. Concepts de graphe de base
   - 2.3. Relations de base
   - 2.4. Intervention des fichiers XML
3. Méthode de recherche

Introduction de chapitre

Dans ce chapitre, nous exposerons l'approche du prototype la plus expérimentale, à savoir l'approche selon le formalisme des graphes conceptuels. Nous nous sommes basés sur les travaux du Professeur Gerbé qui, dans sa thèse intitulée *'Un modèle uniforme pour la modélisation et la métamodélisation d'une mémoire d'entreprise'* [GERBE00-1], analyse la capacité du formalisme des graphes conceptuels à répondre au besoin de stockage et de transmission de la connaissance. Le travail effectué consistait essentiellement à valider la théorie exposée dans cette thèse, à savoir si le formalisme répondait bien à cette capacité à stocker et à gérer la connaissance et ce, d'une manière aussi puissante qu'exposée par le Professeur Gerbé.

Nous exposerons brièvement le formalisme des graphes conceptuels et ses notions de base. Ce formalisme très récent a été conçu par Sowa en 1984 et se base sur les graphes existentiels de Peirce. Cette théorie étant très récente, elle fait aujourd'hui l'objet de nombreuse études et recherches. Une standardisation de cette théorie est aujourd'hui en cours. [SOWA01]. Nous décrirons donc les notions de concept et de relation, la hiérarchie de types ainsi que des règles de dérivation concernant les graphes.

Nous verrons en quoi ce formalisme permet de répondre efficacement aux besoins des Systèmes de Gestion de Mémoire d'Entreprise (SGME). Ces besoins définis dans la thèse précédemment citée du Professeur Gerbé sont au nombre de trois : la classification et la connaissance partielle, les relations entre catégories et/ou instance ainsi que les relations entre catégories ou les instances et le métamodèle. Cette section passera sous la loupe trois formalismes parmi les plus courants qui sont le modèle Entité-Association, un formalisme orienté objet (*Unified Modeling Langage*) et le formalisme des graphes conceptuels. Nous analyserons pour ces trois formalismes leur capacité à combler les besoins précités.

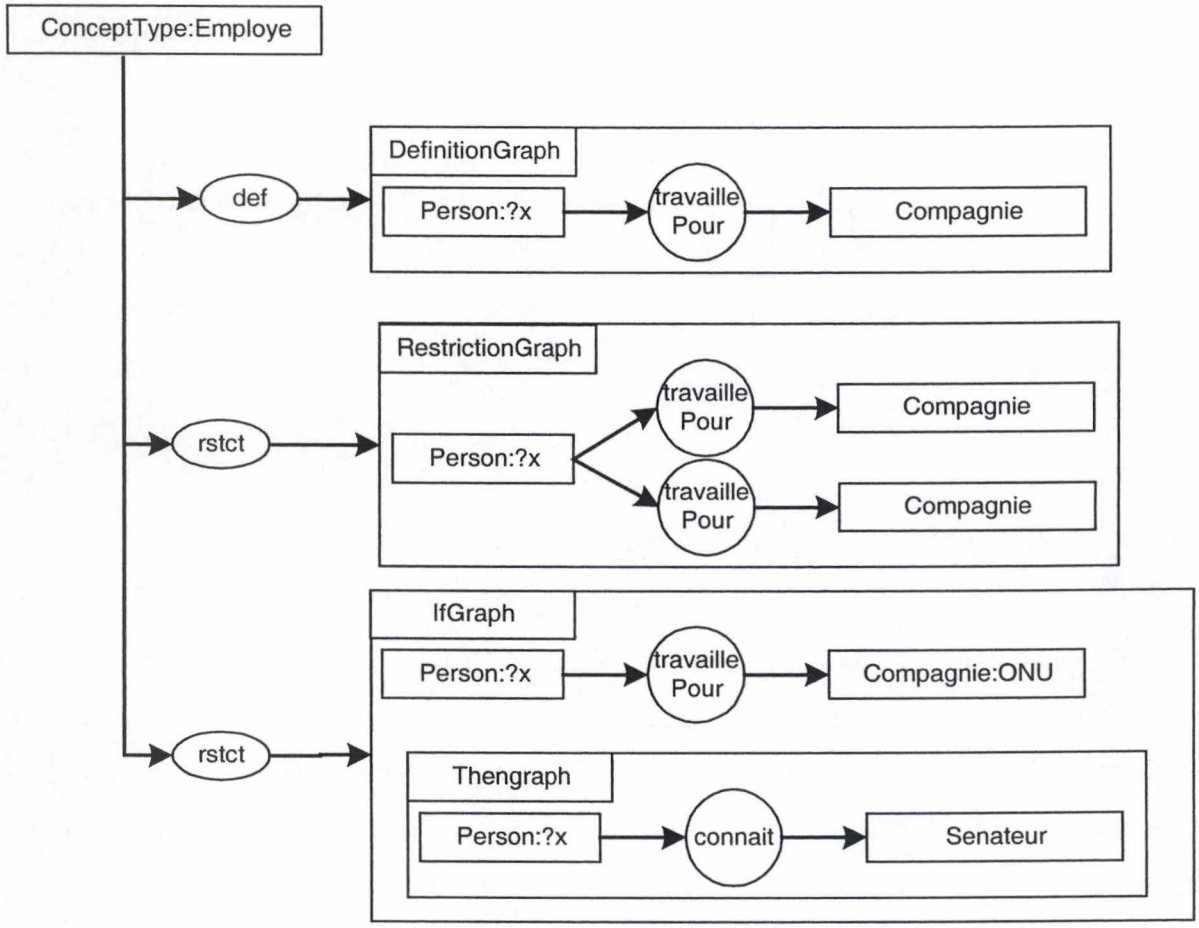
Après avoir vu que ce formalisme était le plus apte à répondre à ces trois besoins, selon la thèse du Professeur Gerbé, nous analyserons une manière efficace de représenter la connaissance par les graphes conceptuels. Cette théorie est très récente. De ce fait, les conventions de représentation diffèrent dans la littérature et ce malgré une standardisation progressive. Nous commenterons un métamodèle de présentation tiré de l'article

'Conceptual graphs, metamodeling and notations of concepts' du Professeur Gerbé qui mettra en lumière « an obvious need for a complete theory of metamodeling in the CG formalism. », CG signifiant Conceptual Graph. [GERBE00-2, p15].

Nous analyserons le prototype implémenté pour tester la capacité des graphes conceptuels à répondre aux besoins des SGME. Nous analyserons les différentes fonctionnalités que nous avons mises en place, nous verrons son fonctionnement et exposerons brièvement l'outil de recherche dont nous avons collaboré à l'élaboration.

I. Introduction aux graphes conceptuels

Nous allons décrire brièvement dans cette partie le fonctionnement des graphes conceptuels. Nous allons présenter quelques notions de base et la manière dont on les représente. Nous irons plus en avant dans l'exposé de ce formalisme et nous verrons que les graphes conceptuels peuvent être représentés sous d'autres formes que la forme graphique. De plus, nous verrons les différentes opérations qui peuvent y être appliquées. Celles-ci permettront de comprendre cette approche du prototype Anthémis. Nous allons tout de suite donner un exemple de graphe conceptuel pour bien situer sa structure. Ses composantes s'expliqueront au fil de ce chapitre.



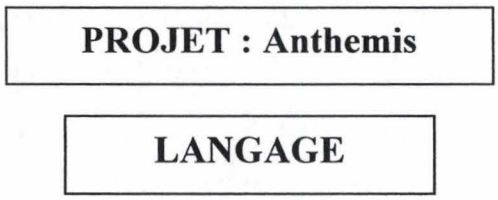
SCHEMA 7.1 : Exemple de graphe conceptuel.

1. Notion de concept et de relation

La théorie des graphes conceptuels est un nouveau formalisme qui a vu le jour en 1984. Nous devons le développement de cette théorie à Sowa qui l'a introduite dans son livre *'Conceptual Structures : Information Processing in Mind and Machine'* [SOWA84]. Ce formalisme très simple se rapproche fortement de la langue naturelle. Il est compris aussi bien par l'homme que par les machines. De plus, il facilite l'échange de données via le protocole Conceptual Graph Interchange Form (CGIF).

Un graphe conceptuel, comme son nom l'indique, se compose en premier lieu de concepts. Ceux-ci peuvent être soit individuels soit génériques et ils représentent les objets

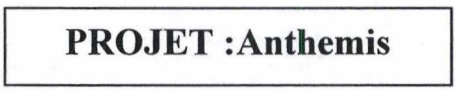
de l'univers du discours. Ces concepts sont représentés sous forme graphique par des rectangles. (SCHEMA 7.2)



SCHEMA 7.2 : Deux concepts.

Nous avons deux concepts qui sont respectivement le PROJET Anthemis ainsi que le concept de LANGAGE. Nous pouvons remarquer que le PROJET Anthemis est un concept individuel tandis que le concept de LANGAGE est un concept générique.

Un concept individuel est unique, c'est-à-dire qu'il est associé à un référent. Il est la représentation d'un objet déterminé. Ce concept individuel se représente sous forme graphique par un rectangle qui contient aussi bien le référent du concept individuel que son type. L'exemple suivant illustre un tel concept. Le référent est Anthemis qui est de type PROJET. Ce concept représente donc le projet Anthemis. (SCHEMA 7.3)



SCHEMA 7.3 : Un concept individuel.

Un concept générique, contrairement aux concepts individuels, représente un objet indéterminé. Celui-ci désigne en quelque sorte un objet d'une catégorie particulière et se représente simplement par un rectangle contenant son type. Dans l'exemple suivant, nous avons affaire à un concept générique représentant *un* PROJET. (SCHEMA 7.4)

SCHEMA 7.4 : Un concept générique.

Maintenant que nous avons vu les différents concepts, il est nécessaire d'agencer ces derniers de manière à pouvoir exprimer une phrase, une idée, une connaissance. Pour cela, nous introduisons les relations conceptuelles. Celles-ci sont représentées par des arcs reliant deux concepts. Ces relations peuvent être définies en ayant un référent ainsi qu'un type au même titre que les concepts. Il est important de noter que la plupart des relations sont binaires comme le montre l'exemple suivant : la relation DIRIGE relie deux concepts qui sont Gerbé et Anthemis qui ont respectivement le type professeur et le type projet. (SCHEMA 7.5)

SCHEMA 7.5 : Notion de relation conceptuelle.

2. Hiérarchie des types

Comme nous l'avons vu et comme dans la plupart des formalismes, chaque concept peut être associé à un type. De la même manière, chaque relation peut elle-même posséder un type. Tous ces types peuvent être aménagés de façon hiérarchique de la manière suivante : la hiérarchie de type est un ordre partiel qui possède un type universel noté $\top$ ainsi qu'un type absurde noté $\bot$ de telle sorte que tout concept est de type universel et a contrario, il n'existe aucun concept de type absurde mais chaque concept est sur-type de ce type. Ces deux types servent donc à fixer les extrêmes de la hiérarchie c'est à dire que dans chaque hiérarchie de type, il y aura au sommet le type universel et à la base le type absurde.

Selon Sowa, « the type hierarchy is a partial ordering defined over the set of types labels. The symbol $\leq$ designates the ordering. Let s, t and u be type labels :

- If $s \leq t$, then s is called a proper subtype of t and t is called a supertype of s, written $t \geq s$.
- If $s \leq t$ and $s \neq t$, then s is called a proper subtype of t, written $s < t$; and t is called a proper supertype of s, written $t > s$.

- If s is a subtype of t and a subtype of u ($s \leq t$ and $s \leq u$) , then s is called a common subtype of t and u.
- If s is a supertype of u ($s \geq t$ and $s \geq u$), then s is called a common supertype of t and u. » [SOWA84]

Le type universel sera notamment utile lors de l'élaboration de l'outil de recherche dans la base de connaissances.

3. Forme logique et linéaire

Comme nous l'avons déjà remarqué précédemment dans le troisième chapitre, le formalisme des graphes conceptuels est très proche de la langue naturelle. Les graphes conceptuels peuvent être aisément traduits dans une autre forme logique comme la logique des prédicats. Si l'on prend l'exemple ci-dessous,

Sa forme logique sera :

$$(\exists x)\ (\exists y)\ ((\text{PROFESSEUR}(x) \wedge \text{PROJET}(y) \wedge \text{DIRIGE}(x,y))$$

Selon Sowa, il existe une méthode de transformation des graphes conceptuels en une forme logique qui utilise un opérateur particulier désigné sous la lettre grecque Φ. Si u est un graphe conceptuel, alors Φu est une formule déterminée par les étapes suivantes :

- Si u contient k concepts génériques, on assigne une variable distinctes $x_1,....,x_k$ à chacun de ces concepts.
- Pour chaque concept c de u, identifiant(c) est la variable assignée à c si celui-ci est générique ou référent(c) si celui-ci est individuel.
- On représente chaque concept c par un prédicat dont le nom est le même que le type de c et avec comme argument identifiant(c).

- On représente chaque relation conceptuelle *r* par un prédicat dont le nom est le même que le type de *r* et les arguments sont les identifiants de chaque concepts reliés par cette relation conceptuelle.
- Enfin, on rajoute les quantificateurs pour chaque variable $x_1,...,x_k$: $\exists x_1...\exists x_k$ et le corps est composé de la conjonction de tous les prédicats associés aux concepts et aux relations. [SOWA84, p86]

De même, les graphes conceptuels peuvent être représentés sous une forme linéaire dans laquelle les concepts sont représentés entre crochets et les relations entre parenthèses. Les arcs sont, quant à eux, représentés par des flèches orientées.

[PROFESSEUR : Gerbé] → (DIRIGE) →[PROJET : Anthemis]

4. Opérations de dérivation

Nous allons maintenant présenter quelques opérations et définitions de base pour la dérivation et la structuration des graphes conceptuels. Ces opérations seront utiles par la suite lors de la description du prototype selon l'approche des graphes conceptuels.

Il existe quatre opérations de dérivation qui sont les suivantes : la simplification, la restriction de concept, la restriction de relation et la jointure.

Premièrement, la simplification est le fait que si deux relations conceptuelles sont identiques dans un graphe, une des deux relations peut être supprimée ainsi que ses arcs.

Deuxièmement, la restriction de concept signifie qu'un concept peut être restreint en remplaçant son type par un type plus spécialisé, c'est-à-dire un sous-type. De même, on peut remplacer un concept générique représentant une entité anonyme par un concept individuel représentant une entité identifiée.

Troisièmement, la restriction de relation suit le même principe que la restriction de concept, à savoir qu'une relation peut être remplacée par une de ses sous-types.

Quatrièmement, le principe de la jointure est le suivant : si un concept *c* du graphe *g* est identique à un concept *d* du graphe *v*, le graphe *w* est le résultat de l'union de *g* et *v* en supprimant *d* et en reliant les arcs de *d* à *c*.

5. Graphes conceptuels canons et généralisation

5.1. Graphes conceptuels canons

Nous allons maintenant présenter les définitions inhérentes aux graphes conceptuels canons ainsi qu'à leur dérivation tirées de l'ouvrage de Sowa précédemment cité.

La première définition est la suivante :

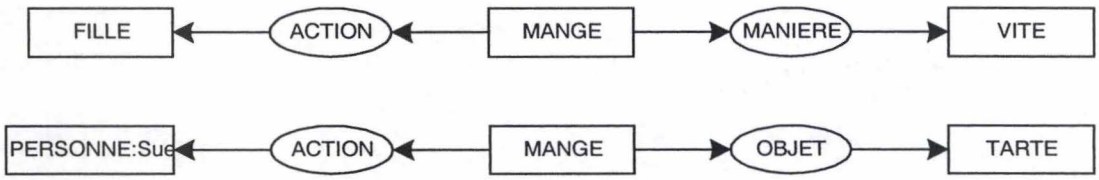
> Un graphe *w* est dit canoniquement dérivable d'un ensemble de graphes A si une des deux conditions suivantes est remplie :
> - *w* est un élément de A,
> - *w* est un graphe dérivé de graphes eux-mêmes canoniquement dérivables de A.
>
> [SOWA84, p96]

Certains graphes sont dits canoniques et d'autres peuvent le devenir selon trois procédés décrit par Sowa :

- Perception : Any conceptual graph constructed by the assembler in matching a sensory icon is canonical.
- Formation rules : new canonical graphs may be derived from other canonical graph by the rules copy, restrict, join and simplify.
- Insight : arbitrary conceptual graphs may be assumed as canonical. [SOWA84, p91]

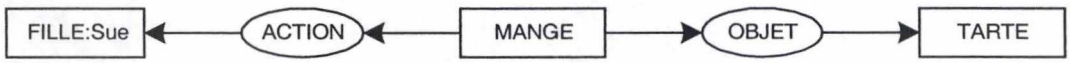
Nous allons maintenant présenter ce deuxième procédé qui correspond à l'obtention de graphes canoniques par les quatre opérations de dérivation présentées plus haut. Les deux autres procédés ne seront pas abordés dans ce travail. Nous allons illustrer nos propos par un exemple très simple tiré du même ouvrage de Sowa :

Si nous avons deux graphes canoniques :



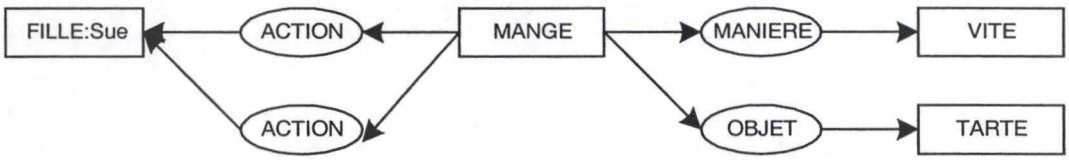
SCHEMA 7.6 : Deux graphes canoniques.

Si nous appliquons la restriction, seul le deuxième graphe change :



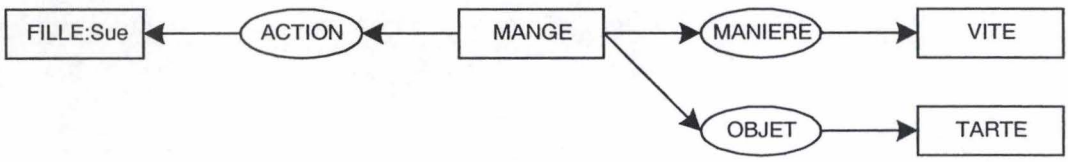
SCHEMA 7.7 : Application de la règle de restriction.

Si nous appliquons le *join*, nous obtenons le graphe suivant :



SCHEMA 7.8 : Application de la règle *join.*

Enfin, en appliquant la simplification, on obtient le graphe canon suivant qui signifie : *Une fille, Sue, mange une tarte rapidement.*



SCHEMA 7.9 : Application de la règle de simplification.[SOWA84, p92-93]

La seconde définition est la suivante :

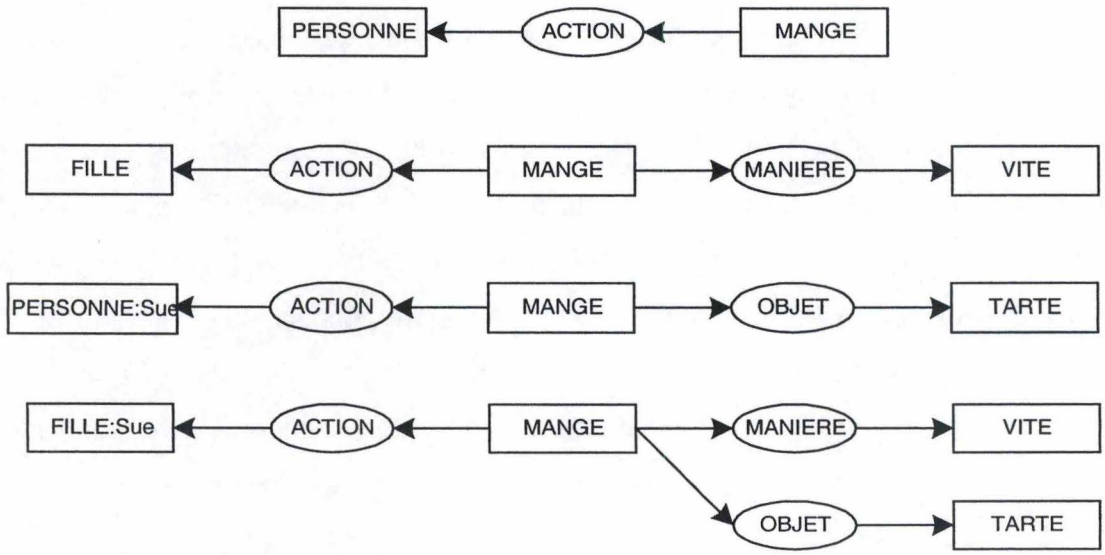
> Si un graphe u est canoniquement dérivable d'un graphe v, u est dit une spécialisation de v notée $u \leq v$, et v est dit une généralisation de u.
> [SOWA84, p97]

5.2. Généralisation

Théorème : La généralisation définit un ordre partiel sur les graphes conceptuels, c'est-à-dire une hiérarchie de généralisation. Pour tout graphe u, v et w, on a les propriétés suivantes :

- Réflexive : $u \leq u$
- Transitive : si $u \leq v$ et $v \leq w$ alors $u \leq w$
- Antisymétrique : si $u \leq v$ et $v \leq u$ alors $u = v$
- Sous graphe : si u est un sous graphe de v alors $v \leq u$
- Sous-type : si u est identique à v excepté que des concepts de v ont été restreints à des sous-types dans u alors $u \leq v$. [SOWA84, p97]

Illustrons ces propos par un exemple lui aussi tiré de Sowa. Le premier graphe est une généralisation des deux suivants. Quant au dernier, il est une spécialisation des précédents.



SCHEMA 7.10 : Spécialisation et généralisation de graphes.

II. Le choix d'un formalisme

1. Introduction

Le but de cette section est de justifier l'emploi de la théorie des graphes conceptuels pour développer un modèle permettant de gérer la connaissance. Nous nous basons essentiellement sur la thèse du Professeur Gerbé [GERBE00-1]. L'étude qu'il a menée sur les formalismes de structure a permis de mettre en lumière la puissance du formalisme des graphes conceptuels dans l'élaboration d'un métamodèle pour la gestion d'une mémoire d'entreprise.

Cette étude débute par la définition de trois besoins fondamentaux pour la structuration des données. Ensuite, trois formalismes parmi les plus courants sont analysés et leur capacité à répondre à ces besoins fondamentaux est testée.

2. Les besoins fondamentaux

Examinons tout d'abord les trois besoins essentiels pour la structuration des données qui sont : la classification et la connaissance partielle, les relations entre catégorie et/ou instance ainsi que les relations entre catégories ou instances et le métamodèle.

La classification est un problème important dans les bases de données relationnelles. En effet, des catégories sont créées à partir des propriétés communes de leur instance. Ces catégories sont basées sur des critères. Mais si le nombre de critères augmente, le nombre de catégories augmente lui de façon exponentielle. De plus, il est difficile de classer un objet dont toutes les propriétés ne sont pas connues. Dès lors, « pour supporter les besoins de classification dans un contexte de connaissance partielle, le formalisme doit, du point de vue de l'utilisateur, supporter la multi-classification, c'est-à-dire qu'un objet peut être défini comme une instance de plus d'une catégorie, et le système doit être capable de faire migrer dynamiquement les instances d'une catégorie à une autre catégorie. ». [GERBE00-1, p22-23]

Dans un formalisme adéquat, on doit être capable de créer des relations entre des niveaux différents de modélisation et en particulier entre une catégorie et une instance. Pour reprendre un exemple du Professeur Gerbé, si nous voulons définir qu'un *employé international* est une *personne* qui *travaille pour l'ONU,* il faut créer un type définissant une relation avec l'organisation ONU qui est une instance de *Compagnie.* Dans le cas d'un modèle Entité–Association, il faut créer un nouveau type qui n'aura qu'une seule instance. Ainsi, « Le formalisme doit supporter les relations entre catégories et/ou instances, c'est-à-dire qu'une catégorie peut être liée à une autre catégorie ou à une instance. ». [GERBE00-1, p24]

Le Professeur Gerbé cherche à donner une description formelle d'objets et de notions permettant d'élaborer un modèle de connaissances. Ce modèle utilise certains composants et certaines relations nécessaires pour représenter cette connaissance. Le métamodèle cherche alors à donner une description de ces composants et de leurs relations. Les catégories fournissant ces informations sur les instances doivent elles aussi être vues comme des instances dans un métamodèle. En effet, « pour supporter ce besoin, le formalisme doit permettre qu'un élément soit vu comme une catégorie ou comme une instance. ». [GERBE00-1, p26]. Le métamodèle que nous verrons dans la troisième partie de ce chapitre se présente comme un ensemble d'éléments permettant de représenter les connaissances. Celui-ci permet à l'utilisateur d'insérer son propre modèle de connaissances en suivant le métamodèle. Le métamodèle contient aussi des éléments qui permettent de vérifier la validité syntaxique de la connaissance comme les graphes de définition qui seront abordés plus loin. Ce besoin exige du formalisme qu'il puisse gérer plusieurs niveaux de modélisation. Le niveau données qui est le premier contient les instances des catégories. Ces catégories sont contenues dans le deuxième niveau et définissent la structure de la connaissance. Ces catégories doivent pouvoir être vues aussi comme des instances du métamodèle contenu dans le troisième niveau. Ce niveau gère alors les différentes structures, les différents modèles pouvant contenir la connaissance.

3. Les formalismes étudiés

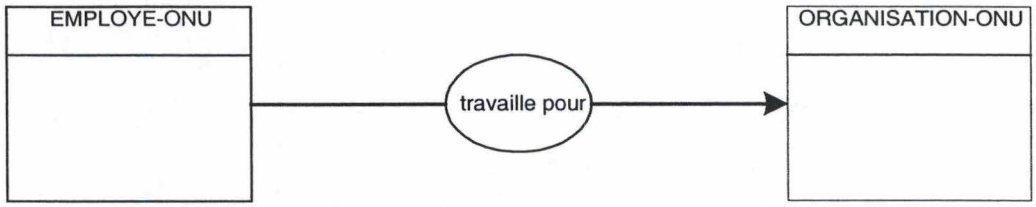
Les cinq formalismes étudiés dans la thèse du Professeur Gerbé sont le modèle Entité-Association étendu, un formalisme orienté objet (UML), le modèle relationnel, un système de la famille KL-ONE appelé Classic et la théorie des graphes conceptuels. Nous ne nous attarderons pas sur Classic, car cela dépasserait le cadre du présent travail. De même pour le modèle relationnel, nous considérons qu'il est très proche du modèle Entité-Association et nous en faisons donc l'économie. Concentrons nous essentiellement sur le modèle Entité-Association présent dans la première approche, sur le formalisme orienté objet présent dans la méthode REX ainsi que sur les graphes conceptuels.

3.1. Modèle Entité-Association

Le modèle Entité-Association a été présenté dans le sixième chapitre, il ne nous paraît pas utile d'en représenter les bases et les principes. Précisons simplement que celui-ci développe la notion de catégorie que l'on appelle les entités ainsi que les relations.

La première constatation qui paraît évidente est qu'il n'est pas possible de représenter les instances dans ce formalisme. Dès lors, « il n'est prévu aucun mécanisme pour la migration des instances et pour représenter la connaissance partielle. ». [GERBE00-1, p26]

Comme nous l'avons vu plus haut, la seule façon de répondre au besoin d'une relation entre catégories et/ou instances est de « créer une entité qui n'aura qu'une seule instance, de définir la relation entre les entités et d'ajouter une note explicative. ». [GERBE00-1, p27] (SCHEMA 7.11)



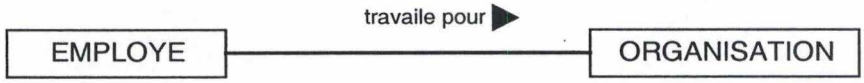
Note : ORGANISATION-ONU a une seule instance ONU

SCHEMA 7.11 : Relation entre catégories et/ou instances. [GERBE00-1 p27]

Pour le troisième besoin constitué par les relations entre les catégories ou les instances et le métamodèle, on constate de nouveau que le modèle Entité-Association ne supporte qu'un seul niveau. Il faut donc user d'une note explicative pour combler ce besoin.

3.2. UML

Comme formalisme orienté objet, le Professeur Gerbé s'est intéressé à Unified Modeling Langage (UML) qui a été développé par Booch, Rumbaugh et Jacobson.[10] « Les instances sont des objets et les catégories des classes et la classification est supportée. Dans la notation UML les classes sont supportées par des boîtes rectangulaires, les instances sont représentées comme des classes, mais avec un libellé souligné. Les relations sont représentées par des arcs joignant les classes .». [GERBE00-1, p27-28] (SCHEMA 7.12)



SCHEMA 7.12 : Exemple UML.

Ce formalisme supporte donc la classification et permet l'utilisation de valeurs nulles pour le support de la connaissance partielle. Ce procédé permet de répondre au premier besoin, la classification et la connaissance partielle.

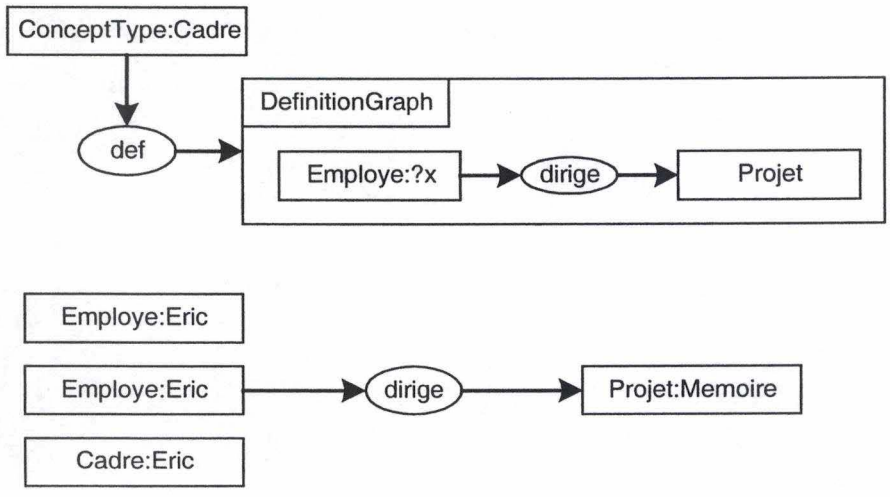
[10] G.Booch, J.Rumbaugh, I.Jacobson, *Unified Modeling Langage, Version 1.1, 1997.*

En ce qui concerne le besoin de relation entre catégories et/ou instance, la solution est la même que pour le modèle Entité-Association, à savoir l'ajout d'une note explicative. De même, pour les catégories et les instances, leur représentation étant différente, il n'est pas possible de dire qu'une instance est une classe.

3.3. Graphes conceptuels

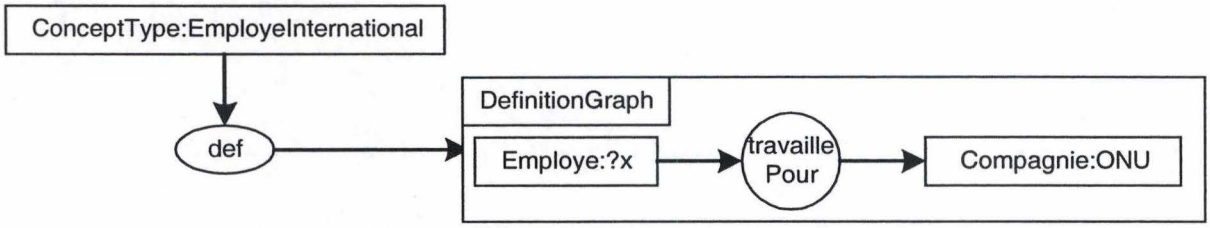
En ce qui concerne le premier besoin, deux conditions doivent être satisfaites : la classification ainsi que la connaissance partielle. Comme nous l'avons vu dans le troisième chapitre, les graphes conceptuels introduisent une hiérarchie de type balisée par deux types extrêmes qui sont le type universel et le type absurde. Grâce à cette hiérarchie, on peut aisément classer les connaissances.

Le formalisme des graphes conceptuels répond très bien aussi à l'insertion de connaissance partielle. En effet, la hiérarchie des types permet la migration d'un référent d'un type vers un autre au fur et à mesure que les informations se complètent. Dans l'exemple suivant, on voit l'employé Eric qui migre en tant que cadre alors qu'il est assigné à un projet. En effet, le premier graphe montre qu'un cadre est une personne qui dirige un projet. Tout d'abord, Eric est un employé et ensuite un projet va lui être assigné et il va le diriger. Enfin, Eric migrera en temps que cadre grâce à cette nouvelle conaissance.



SCHEMA 7.13 : Gestion de la connaissance partielle dans les graphes conceptuels.

Le deuxième besoin concerne la relation entre les catégories et les instances, c'est-à-dire que l'on doit être capable de mettre en relation deux instances mais aussi une instance avec une catégorie. Dans le schéma suivant on constate que le problème posé par le formalisme Entité-Association est bien résolu en utilisant les graphes conceptuels. En effet, le type *employé international* sera définit comme suit :



SCHEMA 7.14 : Relation entre catégorie et instance dans les graphes conceptuels.

On peut remarquer que la relation se fait entre une catégorie et une instance, la catégorie étant *Employe* et l'instance de type *Compagnie* qui est *ONU*.

Le dernier besoin concernant les catégories ou les instances dans le métamodèle est lui aussi bien rempli par le formalisme des graphes conceptuels. En effet, si l'on prend le type organisation qui est de type concept, ce type peut être une catégorie d'une instance qui est ONU. Donc, le type organisation peut être à la fois une catégorie pour l'instance ONU ou une instance du type Concept et cela à différents niveaux, dans le métamodèle et dans le modèle.

En conclusion, selon le Professeur Gerbé, « aucun formalisme ne propose une couverture complète des besoins associés à la modélisation et à la métamodélisation d'une mémoire d'entreprise. Cependant, nous avons vu que le formalisme des graphes conceptuels offre la puissance d'expression et la flexibilité requises ». [GERBE00-1, p49] Notons que cette conclusion n'engage que son auteur et que le but du prototype était de tester effectivement cette théorie. Nous verrons plus loin que cette puissance des graphes conceptuels est vérifiée notamment dans un outil de recherche que nous avons contribué à développer.

	Classification et connaissance partielle	Relations entre catégories et/ou instances	Catégorie ou instance
Entité-Association étendu	Non	Non	Non
Orienté objet	Oui	Non	Non
Graphes Conceptuels	Oui	Oui	Oui

TABLEAU 7.1 : Résumé de la capacité des formalismes de répondre aux besoins fondamentaux.

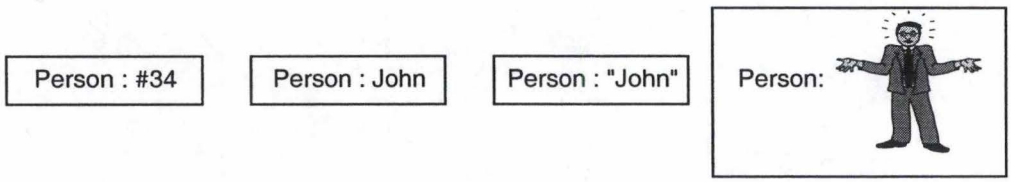
III. Un métamodèle pour la représentation des connaissances

1. Introduction

Maintenant que nous nous sommes quelque peu familiarisés avec les graphes conceptuels et que nous avons vu l'intérêt de choisir ce formalisme pour représenter des connaissances, nous allons examiner le métamodèle de cette représentation. En effet, en vue de représenter les connaissances, nous allons définir un métamodèle général qui permettra de représenter tout type de connaissances. Ce métamodèle permettra aussi de rentrer ses propres structures. Mais la théorie des graphes conceptuels étant très récente, les études sur le sujet ne sont pas encore très nombreuses et le domaine est en plein développement. Cette situation est notamment due à l'absence de standardisation même si celle-ci est en cours. [SOWA01]. Cette section s'inspirera de la thèse du Professeur Gerbé et d'autres publications en la matière. Nous allons essayer de mieux appréhender la vision du prototype.

Comme nous venons de l'indiquer, certains aspects des graphes conceptuels restent très ambigus, ceci étant dû principalement à l'absence de spécification. Nous entendons par là qu'aucune standardisation n'est validée et que la notation des différents

concepts diffère d'une littérature à l'autre. Comme nous l'avons vu dans le début de ce chapitre, les notions de base sont bien définies mais la manière de les noter et de les représenter n'est pas uniforme. Si nous reprenons l'exemple présenté dans *'Conceptuel graphs, metamodeling and notation of concepts'* [GERBE00-2], il existe différentes manières de représenter un concept en utilisant la théorie des graphes conceptuels :

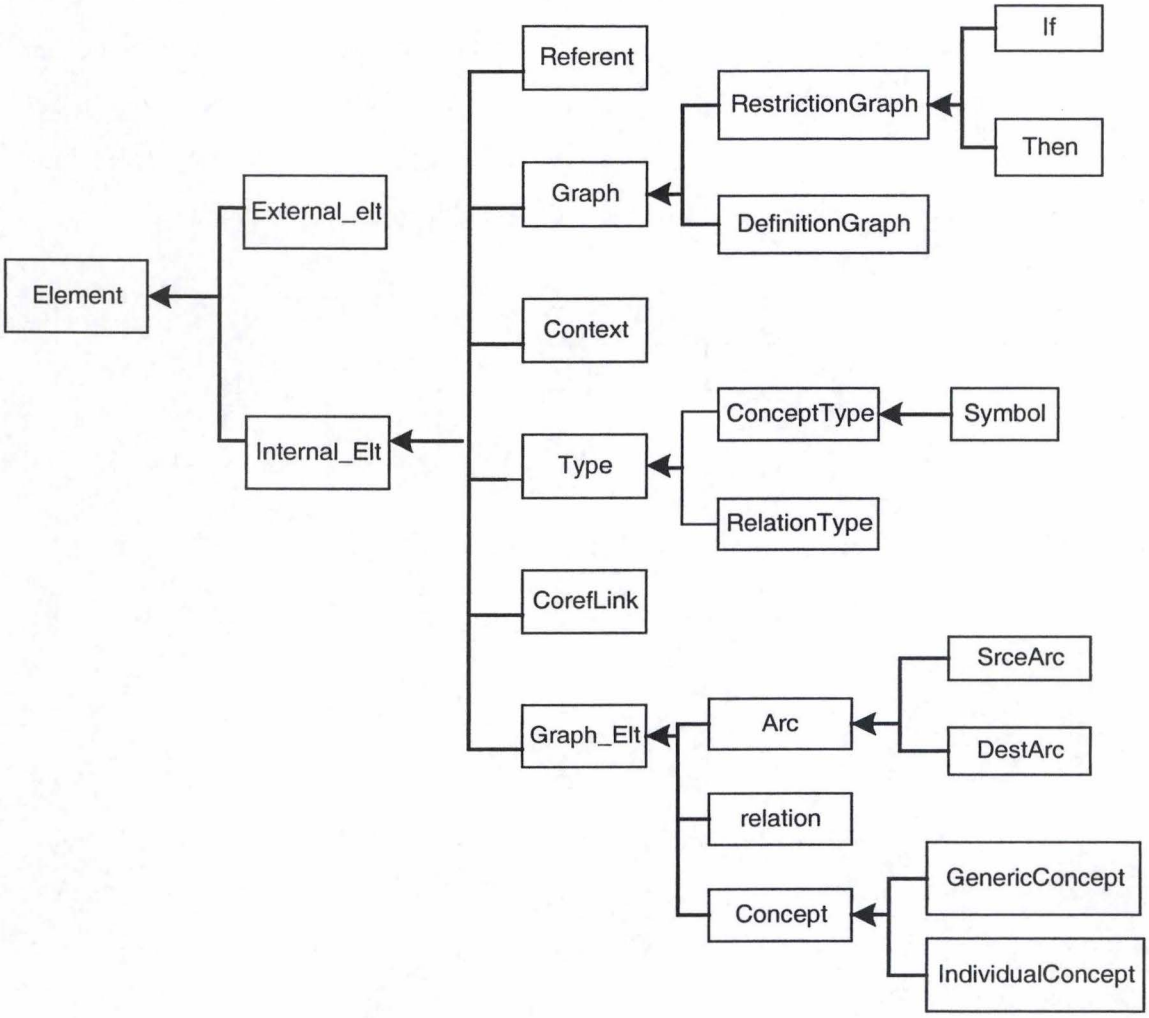


SCHEMA 7.15 : Les différentes façons de représenter un concept. [GERBE00-2, p2]

2. Le métamodèle

Nous allons maintenant présenter un métamodèle permettant de représenter les connaissances, de les manipuler et de les décrire. (SCHEMA 7.16) Nous allons ainsi comprendre qu'il existe « an obvious need for a complete theory of metamodeling in the CG formalism ». [GERBE00-2, p15]

Dans le métamodèle, les éléments se trouvant au niveau le plus élevé sont les éléments externes et les éléments internes. Les éléments externes sont les parties du monde réel que l'on représente. Les éléments internes sont des composants des graphes proprement dits qui permettent de représenter ces éléments externes. Ceux qui nous intéressent ici sont les éléments internes car ce sont eux qui composent les différentes parties des graphes conceptuels. Au niveau inférieur se trouvent les référents, les graphes, les contextes, les types, les éléments du graphe à proprement parler et les liens de co-références. Ces derniers dépassant le cadre de notre exposé, nous ne nous y attarderons pas davantage. Notons simplement que ceux-ci permettent d'associer des éléments représentant la même entité.



SCHEMA 7.16 : Le métamodèle de représentation des concepts.

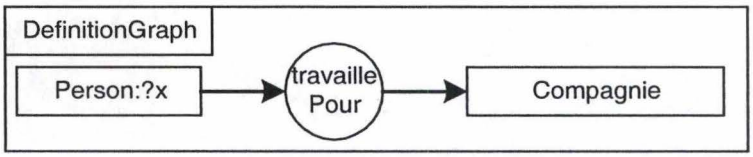
Dans la suite de cette section, nous examinerons tour à tour les graphes, les éléments, les référents ainsi que les concepts. Nous donnerons préalablement une explication des types et nous expliquerons leur fonctionnement.

Les types permettent de définir des catégories contenant des éléments ayant les mêmes propriété. L'élément ConceptType permet donc de définir des types en général. Si nous avons un type de concept qui est *Employé,* celui-ci sera défini grâce à l'élément ConceptType. En d'autres termes, comme nous le verrons dans la troisième section de ce chapitre, l'élément ConceptType est un type de base permettant de définir tous les autres types.

2.1. Les graphes

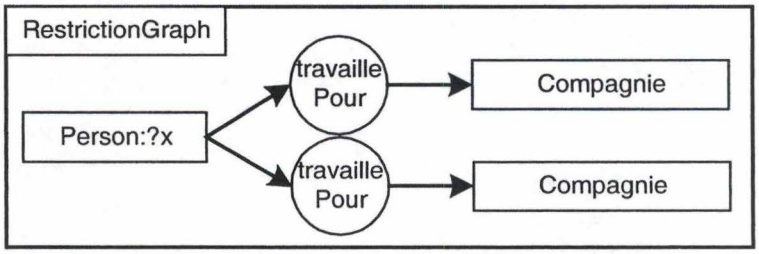
La catégorie des graphes contient les graphes de définition ainsi que les graphes de restriction. Ces derniers se divisent en graphes *if* et *then*.

Les graphes de définition permettent de spécifier le type d'un concept en montrant les relations conceptuelles qu'un concept doit posséder. Le graphe de définition définit en quelques sortes les attributs et les associations du concept bien que ces deux notions n'existent pas dans la théorie des graphes. Ces graphes de définition permettent de vérifier qu'un concept respecte bien les propriétés de son type. De même, lorsque l'on considère la hiérarchie des types, chaque concept doit vérifier le graphe de définition de son propre type mais aussi celui de ses sur-types. Si l'on observe le graphe de définition suivant qui est celui du type *Employé,* c'est une personne qui travaille pour une compagnie.



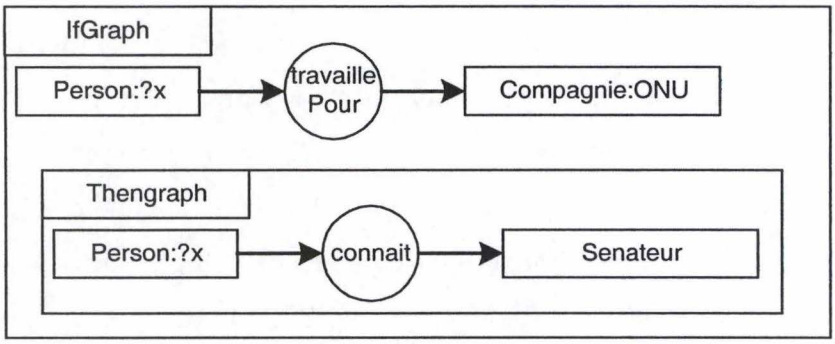
SCHEMA 7.17 : Un graphe de définition.

Les graphes de restriction ont le même fonctionnement que les graphes de définition. Ceux-ci permettent d'exprimer les contraintes qui ne sont pas exprimables via le graphe de définition. Ils représentent entre autres certaines propriétés incompatibles avec leur type mais aussi certaines notions de cardinalité dans les relations des concepts entre eux. Si nous reprenons l'exemple de l'employé, le graphe de restriction nous dit que celui-ci ne peut pas travailler pour deux compagnies.

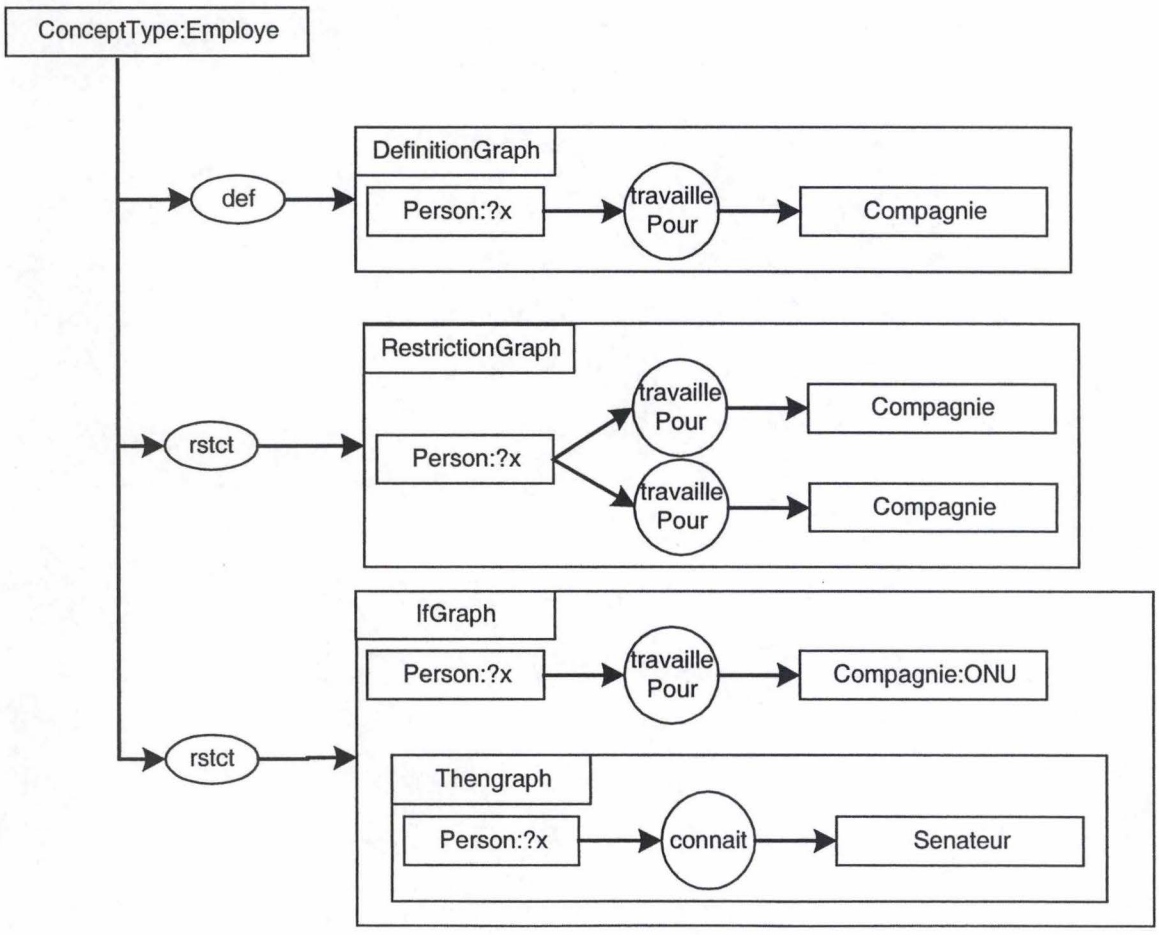


SCHEMA 7.18 : Un graphe de restriction.

Enfin, parmi les graphes de restriction, il existe des graphes un peu plus complexes qui sont les graphes de règles composés de graphes *if* et *then*. Ceux-ci montrent que si une condition est respectée, alors une autre doit l'être aussi. Ainsi, si un employé travaille pour l'ONU, alors la personne connaît un sénateur.



SCHEMA 7.19 : Un graphe *if* et *then*.



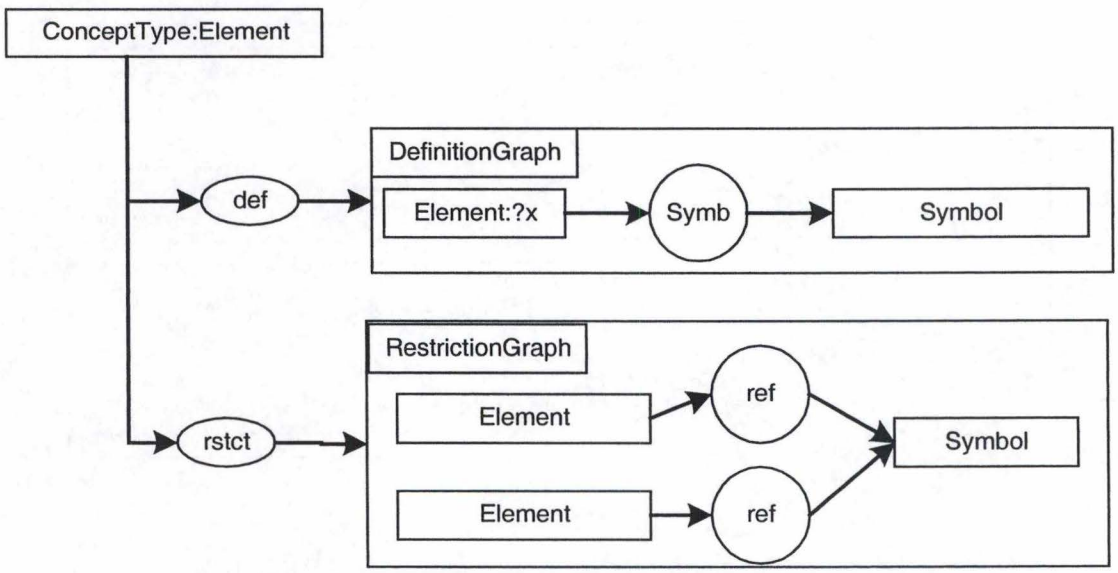
SCHEMA 7.20 : Un graphe de définition complet.

En résumé, ces différents graphes qui sont les graphes de définition, de restriction et de règles permettent de définir les types. Ces graphes permettent de vérifier la validité des concepts introduits et de leur type. De même, ceux-ci permettent aussi de vérifier leur validité des concepts par rapport à leur sur-type.

Après avoir vu les notions de type et de graphe, nous allons maintenant donner une notation et une représentation pour une notion particulière du métamodèle, à savoir les concepts. Pour ce faire, il est utile de définir ce que l'on entend par élément, concept et référent.

2.2. Les éléments

Un graphe conceptuel est composé d'éléments qui sont combinés entre eux pour représenter la connaissance. Un élément peut être vu comme un terme général permettant d'exprimer chaque composante d'un graphe. Chaque élément est associé à un ou plusieurs symboles mais un symbole n'est associé qu'à un et un seul élément. Prenons un élément qui est une personne, celui-ci peut-être symbolisé de différentes manières : un string représentant son nom, un entier symbolisant son numéro de registre national ou bien encore, une photo d'identité.

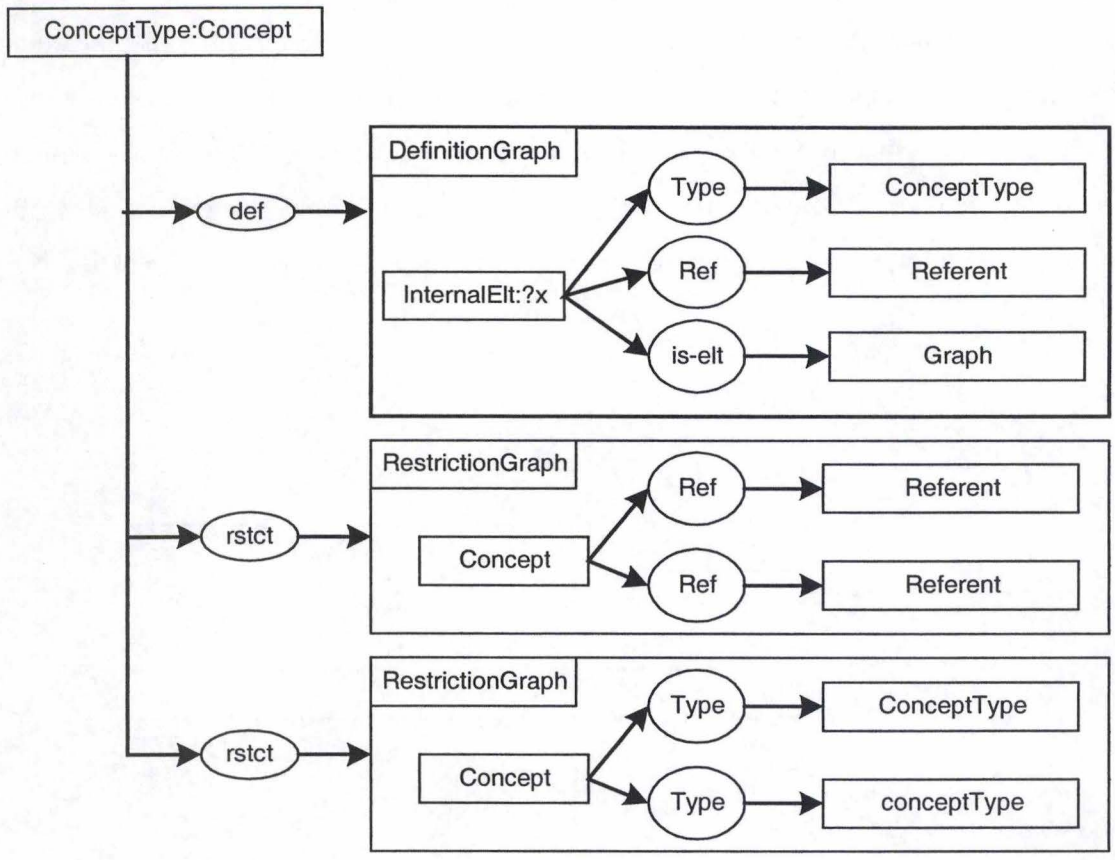


SCHEMA 7.21 : Graphe de définition d'un élément. [GERBE00-2, p5]

2.3. Les concepts

Le concept est l'élément de base de la théorie des graphes conceptuels. Celui-ci peut être défini comme suit : « A concept is the representation of an object of the Universe of Discourse. It is the assembly of two parts : a referrent that identifies the representated object and the type that classifies it ». [GERBE00-2, p6]. Cette définition peut être exprimée sous forme de graphe comme nous l'avons vu plus haut.

Un concept est un élément interne qui possède un type, qui est référencé par un référent et qui appartient à un graphe. Le graphe de restriction montre qu'un concept ne peut être référencé que par un et un seul référent et qu'il n'est que d'un et un seul type. (SCHEMA 7.22)

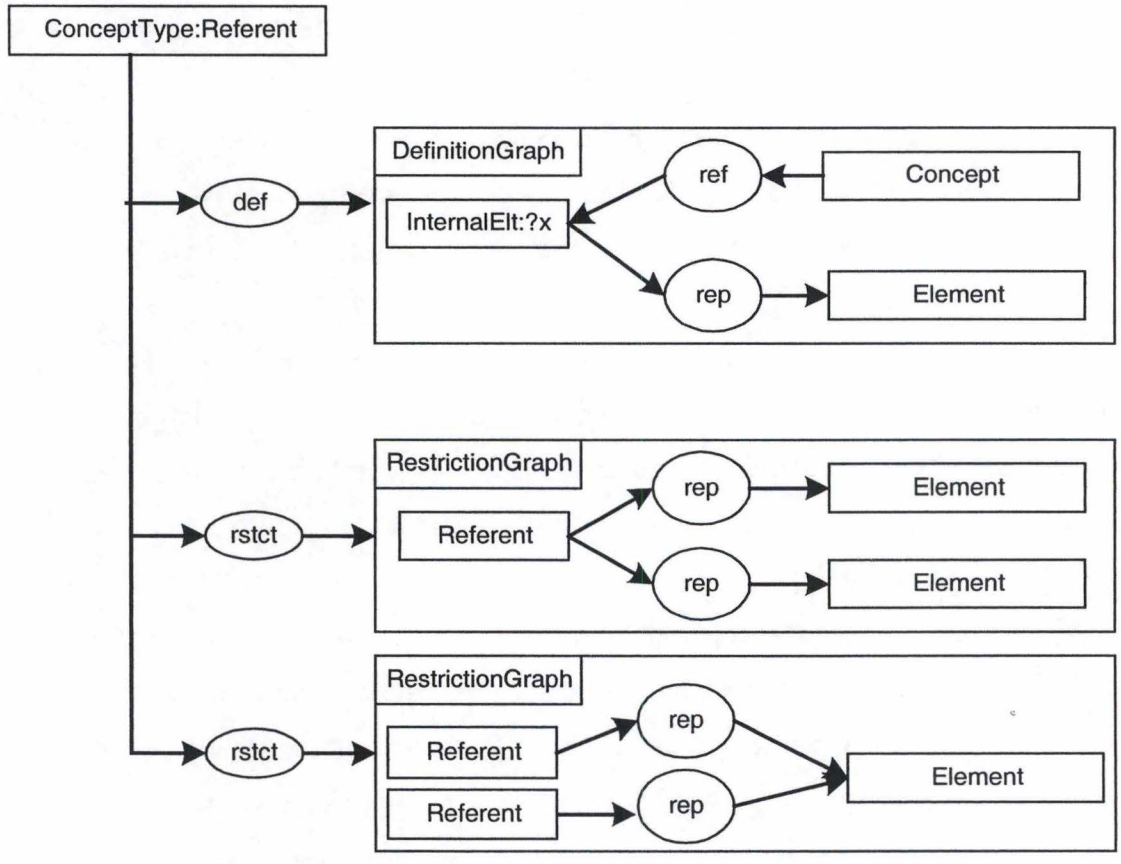


SCHEMA 7.22 : Graphe de définition d'un concept. [GERBE00-2, p6]

Si on reprend un cas vu précédemment, on peut considérer qu'il existe un concept de type Organisation dont le référent est ONU et qui appartient à un graphe particulier. Ce concept ne peut évidemment pas avoir deux référents, son référent étant ONU. De même, ce référent ONU ne peut référencer un autre concept.

2.4. Les référents

Selon le Professeur Gerbé, le réfèrent est la partie du concept qui représente et identifie l'objet de l'univers du discours pour lequel le concept est une interprétation. De la même manière que pour les concepts, un référent peut être définit comme suit : « A referent is a proxy for an object of the universe of the discourse in the knowledge base. A referent is made up of a quantifier and a designator that refers to the object ». [GERBE00-2, p6] (SCHEMA 7.23)



SCHEMA 7.23 : Graphe de définition d'un référent. [GERBE00-2, p7]

Un référent peut être vu comme un élément interne faisant partie d'un concept et qui représente un élément. Les graphes de restriction précisent qu'un référent représente un et un seul élément et que deux référents ne peuvent représenter le même élément. En résumé, un référent sert à désigner un concept particulier, une certaine instance d'un concept générique.

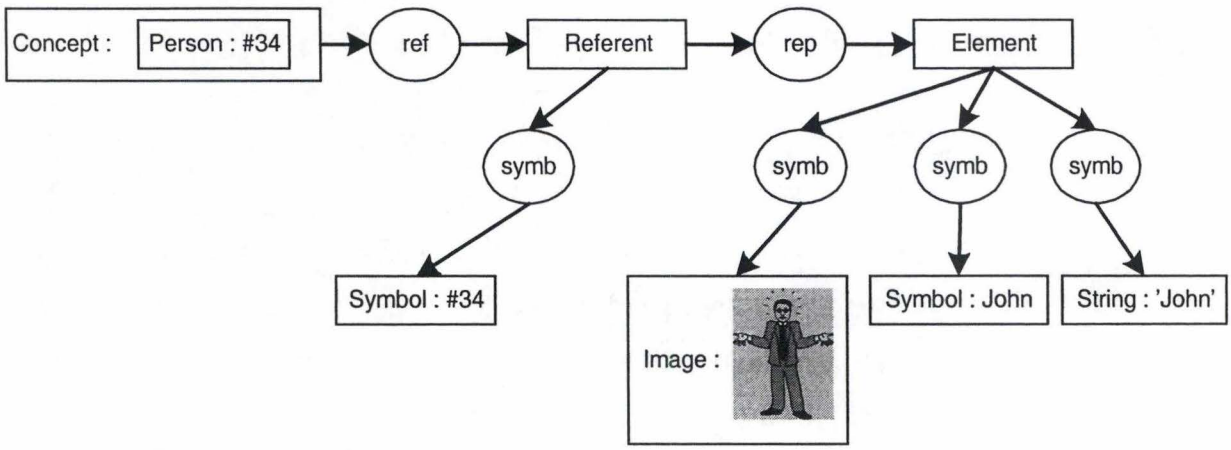
Si l'on reprend notre exemple de l'ONU, le référent ONU référence un concept particulier de type Organisation et représente un élément. C'est-à-dire ici, le référent ONU représente l'élément : l'Organisation des Nations Unies. Cet élément peut être symbolisé de différentes manières comme par l'image d'un drapeau caractéristique ou bien par un string (Organisation des Nations Unies).

2.5. Notation

Nous allons maintenant préciser une notation possible pour ces notions de la théorie des graphes conceptuels. En effet, dans la littérature existante, on trouve beaucoup de notations différentes pour les représenter. Selon le Professeur Gerbé, un concept possède un référent qui représente un élément. Un référent est symbolisé par un symbole, tandis que l'élément peut être symbolisé par une image, un string ou bien un symbole.

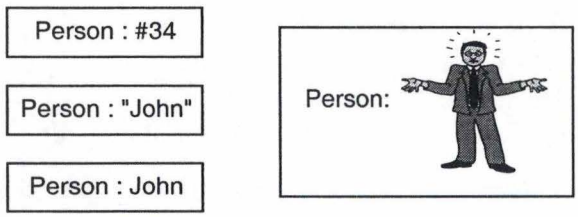
L'exemple suivant montre cette représentation qui associe à un concept (une personne) un référent symbolisé par 34. Le symbole du référent commence toujours par le signe #. Ce même référent représente un élément qui est symbolisé de trois manières différentes : une image, un symbole et un string. (SCHEMA 7.24)

Nous avons une personne particulière qui est référencée par un référent qui est JOHN. Ce référent représente un élément qui est symbolisé de différentes manières. Cet élément que l'on peut appeler la personne John Smith habitant telle adresse, etc. est symbolisé par une image qui est sa photo d'identité, un string qui est son nom John Smith, et un autre symbole qui peut être son numéro de registre national (ou son prénom dans ce cas ci).



SCHEMA 7.24 : Graphe de représentation d'un concept. [GERBE00-2, p9]

Pour posséder une notation commune, le Professeur Gerbé introduit une notation qui est obtenue à partir d'une règle qui est la suivante : chaque référent de concept peut être noté par son propre symbole mais aussi par les symboles de l'élément qu'il représente. Dans notre exemple, nous avons la représentation de la personne :



SCHEMA 7.25 : Notation d'un concept. [GERBE00-2, p10]

Nous venons donc de voir quel métamodèle utiliser dans la théorie des graphes conceptuels pour représenter les connaissances. Ce travail s'est limité à la base, à savoir les concepts, les référents et les éléments. Cela nous permet donc de conclure à la représentation d'un modèle uniforme mais aussi de se rendre compte qu'il est nécessaire de compléter cette métamodélisation, c'est-à-dire de l'étendre aux autres notions vues dans le métamodèle. En effet, nous nous sommes limités dans ce travail aux concepts mais il est nécessaire d'effectuer la même démarche notamment pour les relations. Celles-ci ont besoin de disposer d'une notation uniforme et standard au même titre que les concepts. Notons que l'on retrouve ce complément dans la thèse précédemment citée du Professeur Gerbé.

IV. Anthémis, prototype de gestion de la connaissance par les graphes conceptuels

Il est important de considérer le travail effectué comme un prototype. En effet, le but était de tester la capacité des graphes conceptuels à gérer une mémoire d'entreprise. Il était avant tout primordial de saisir la puissance de ce formalisme. Le travail que nous avons fourni est la création de la base de données, le perfectionnement de l'outil de recherche ainsi que l'implémentation des différentes fonctionnalités permettant de gérer la base de données comme nous l'avons vu dans le chapitre trois.

Contrairement au prototype basé sur le modèle Entité-Association, la base de données du prototype mettant en œuvre les graphes conceptuels ne comporte que deux tables. En effet, comme nous l'avons vu, les graphes conceptuels sont composés de deux notions : les concepts et les relations. Nous aurons donc une table regroupant tous les concepts et une autre contenant les relations.

1. Architecture de la base de données

La table CONCEPTS est composée de trois colonnes que nous allons décrire ici. Premièrement, chaque concept possède un identifiant qui est unique et appelé ID_CONCEPT. Celui-ci suit une séquence prédéfinie appelée SEQ_CONCEPTS. Une séquence SQL permet de donner un identifiant unique à des éléments d'une table, c'est-à-dire qu'elle s'incrémente de un à chaque appel. La deuxième colonne, ID_TYPE, contient le type du concept, c'est-à-dire l'identifiant de son type. Enfin, la troisième colonne, REFERRENT, contient le référent proprement dit du concept qui est un string. Le TABLEAU 7.2. montre un exemple d'une telle table.

ID_CONCEPT	ID_TYPE	REFERENT
1	1	ConceptType
2	2	RelationType
3	1	UniversalType
4	1	Graph
52	1	Personne
53	52	John

TABLEAU 7.2 : La table CONCEPTS.

La table RELATIONS est quant à elle constituée de cinq colonnes. La première colonne reprend l'identifiant de la relation, ID_RELATION, qui est unique et suit la séquence prédéfinie SEQ_RELATIONS. La deuxième colonne est le type de la relation : ID_RELATION_TYPE. Celui-ci est identifié par un nombre qui fait référence à la table concepts. La troisième colonne reprend l'identifiant du graphe auquel appartient la relation : ID_GRAPH. Notons qu'une relation n'appartient qu'à un et un seul graphe alors qu'un concept peut quant à lui appartenir à plusieurs graphes. Les quatrième et cinquième colonnes sont respectivement l'identifiant du concept qui est la source de la relation, ID_CONCEPT_SRCE, et l'identifiant du concept qui est la destination de la relation, ID_CONCEPT_DEST. Le TABLEAU 7.3. représente une telle table.

ID_RELATION	ID_RELATION _TYPE	ID_GRAPH	ID_CONCEPT_ SRCE	ID_CONCEPT_ DEST
1	32	33	6	7
2	..	..	..	..

TABLEAU 7.3 : La table RELATIONS.

2. Insertion et notions de base

Le principe de base de l'insertion des connaissances dans la base de données comprend quatre temps. Premièrement, comme nous l'avons vu, le formalisme des graphes

conceptuels introduit une hiérarchie de concepts. La première opération consiste donc à rentrer cette hiérarchie sous forme de fichier XML. De la même façon, la deuxième opération consiste à introduire dans la base de données la hiérarchie des relations en utilisant le même format. Ensuite, il est nécessaire d'introduire les graphes de définition de ces concepts et de ces relations. En effet, ces graphes permettront de vérifier la validité de chaque connaissance introduite et l'intégrité de la base de données. Chaque connaissance insérée sera confrontée à son graphe de définition et sera validée ou rejetée. C'est cette insertion de connaissance qui constitue la quatrième opération.

2.1. Concepts de base

Pour que ces insertions puissent se faire correctement dans leurs tables respectives, il existe un ensemble de concepts de base ainsi qu'un ensemble de relations de base qui sont créés lors de l'initialisation de la base. Nous allons maintenant examiner ces notions de base et en donner une brève explication.

Tout d'abord aux niveaux des concepts, nous disposons de quatre concepts de base :

- [ConceptType : ConceptType] : c'est le concept initial qui permet de déclarer tous les types de concepts.
- [RelationType : RelationType] : c'est le concept initial qui permet de déclarer tous les types de relations.
- [ConceptType : UniversalType] : c'est le type universel qui est un sur-type de tous les autres types.
- [ConceptType : BlankType] : c'est le type indéfini qui est un sous-type de tous les autres types.

Illustrons nos propos : nous avons le ConceptType personne qui est sur-type des ConceptType homme et femme. UniversalType est un ConceptType sur-type des types personne, homme et femme qui sont eux-mêmes sur-types du type BlankType.

2.2. Concepts de graphe de base

Au niveau des graphes, on retrouve six concepts initiaux :

- [ConceptType : Graph] : concept qui permet de définir un concept en tant que graphe.
- [ConceptType : DataGraph] : c'est le type initial pour définir un graphe de données.
- [ConceptType : DefinitionGraph] : type permettant d'insérer des graphes de définition.
- [ConceptType : RestrictionGraph] : type permettant d'introduire des graphes de restriction.
- [ConceptType : IfGraph] : type permettant d'introduire des graphes *If.*
- [ConceptType : thenGraph] : type permettant d'introduire des graphes *Then.*

2.3. Relations de base

Au niveau des relations, on retrouve quatre relations de base, à savoir :

- [RelationType : blng] : permet de mettre en relation un concept avec le graphe auquel il appartient.
- [RelationType : subt] : permet de mettre en relation un type avec son sur-type et inversement.
- [RelationType : def] : permet de mettre en relation un concept avec son graphe de définition.
- [Relationtype : rstrct] : permet de mettre en relation un concept avec ses graphes de restriction.

Ces différents concepts et relations de base sont initialisés avec la base de données et permettent l'introduction des hiérarchies de concepts et de relations ainsi que des fichiers de connaissance.

Nous allons maintenant à l'aide d'un cas relativement simple, illustrer les quatre étapes d'insertion des connaissances.

```
<concept_type_hierarchy>
        <concept>
        <type>Personne</type>
                <concept>
                        <type>Homme</type>
                </concept>
                <concept>
                        <type>Employe</type>
                </concept>
        </concept>
        <concept>
        <type>Compagnie</type>
        </concept>
</concept_type_hierarchy>
```

Premièrement, nous introduisons la hiérarchie de concepts qui contient le type *personne* qui est un sur-type des concepts *Homme* et *Employe* ainsi que le concept *compagnie*. Remarquons que dans ce simple fichier, nous avons trois sortes de balises prédéfinies : la balise *<concept_type_hierarchy>* qui nous indique que l'on a affaire à une hiérarchie de concepts, la balise *<concept>* qui introduit un concept et à l'intérieur de cette balise, la balise *<type>* qui définit le type du concept. Ces différentes balises s'imbriquent entre elles pour établir la hiérarchie.

Nous allons maintenant introduire de la même façon la hiérarchie de relations :

```
<relation_type_hierarchy>
        <relation>
        <type>Coordonnees</type>
                <relation>
                <type>Residence</type>
                </relation>
                <relation>
                <type>Etat</type>
                </relation>
                <relation>
                <type>Identite</type>
                </relation>
        </relation>
        <relation>
        <type>Possede</type>
        </relation>
        <relation>
        <type>Travaille pour</type>
        </relation>
</relation_type_hierarchy>
```

Nous avons donc la relation *Coordonnees* qui est sur-type des relations *Residence*, *Etat* et *Identite*. De même, nous avons les relations *Possede* et *Travaille Pour*. De la même façon que dans la hiérarchie de concepts, nous avons trois balises qui sont : *<relation_type_hierarchy>, <relation> et <type>.*

Nous pouvons maintenant insérer les graphes de définition qui permettent de vérifier et de conserver l'intégrité de la base de données.

```
<data>
        <concept ref="*x">
            <type>ConceptType</type>
            <referent>Employe</referent>
         </concept>
```

```
//Déclaration du graphe de définition qui est le suivant : une personne travaille pour
//une compagnie
```

```
<concept ref="*def">
        <type>DefinitionGraph</type>
        <referent>def-Employe</referent>
        <concept ref="*p">
                <type>Personne</type>
                <referent/>
        </concept>
        <concept ref="*c">
                <type>Compagnie</type>
                <referent/>
        </concept>
        <relation>
        <code>Travaille Pour</code>
        <srce>?p</srce>
        <dest>?c</dest>
        </relation>
        </concept>
```

//Déclaration du graphe de restriction qui est : une personne travaille pour une //compagnie et cette personne travaille pour une autre compagnie.

```
<concept ref="*rest">
        <type>RestrictionGraph</type>
        <referent>rest-Employe</referent>
                        <concept ref="*pr">
                                <type>Personne</type>
                                <referent/>
                        </concept>
                        <concept ref="*cu">
                                <type>Compagnie</type>
                                <referent/>
                        </concept>
                        <concept ref="*cd">
                                <type>Compagnie</type>
                                <referent/>
                        </concept>
        <relation>
                <code>Travaille Pour</code>
                <srce>?pr</srce>
                <dest>?cu</dest>
        </relation>
        <relation>
                <code>Travaille Pour</code>
                <srce>?pr</srce>
                <dest>?cd</dest>
        </relation>
</concept>
```

//Déclaration des relations du concept Employe avec son graphe de définition et son
//graphe de restriction.

```
        <relation>
                <code>def</code>
                <srce>?x</srce>
                <dest>?def</dest>
        </relation>
        <relation>
                <code>rstrct</code>
                <srce>?x</srce>
                <dest>?rest</dest>
        </relation>
</data>
```

Nous constatons qu'après l'insertion de ce fichier nous avons la définition de *Employe* qui est : c'est une personne qui travaille pour une compagnie mais cette personne ne peut travailler pour deux compagnies différentes. On peut dès lors introduire nos connaissances :

```
<data>
  <concept ref="*x">
    <type>Personne</type>
    <referent>John</referent>
  </concept>
  <concept ref="*p">
    <type>Pays</type>
    <referent>Belgique</referent>
  </concept>
  <concept ref="*c">
    <type>Compagnie</type>
    <referent>Fundp</referent>
  </concept>
  <relation>
    <code>Etat</code>
    <srce>?x</srce>
    <dest>?p</dest>
  </relation>
  <relation>
    <code>Travaille Pour</code>
    <srce>?x</srce>
    <dest>?c</dest>
  </relation>
</data>
```

Nous renvoyons à l'ANNEXE 6 pour un exemple un peu plus détaillé. Dans cette annexe se retrouve aussi chaque fichier temporaire créé avant l'insertion de la connaissance dans la base de données.

2.4. Intervention des fichiers XML

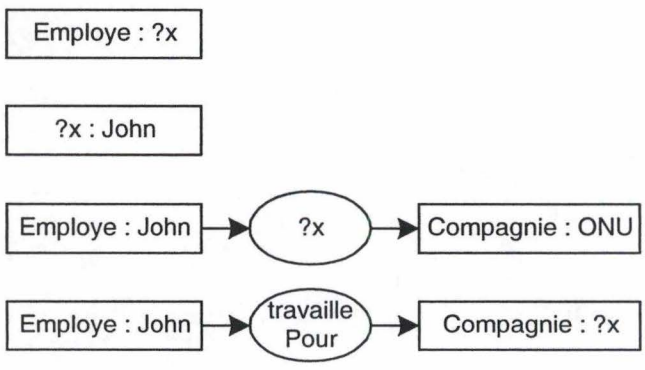
Comme pour l'approche Entité-Association, l'utilisateur doit insérer ses fichiers XML. Un fichier XSL sera appliqué à ses fichiers. Pour une description complète du

mécanisme nous vous renvoyons au sixième chapitre et aux ANNEXES 6 et 7 qui reprennent le code du prototype et en particulier les deux fichiers XSL ainsi qu'un exemple un peu plus conséquent. Le processus reste le même que pour l'approche Entité-Association.

3. Méthode de recherche

Lors de notre arrivée, l'étudiant nous précédant avait déjà commencé l'implémentation d'une méthode de recherche dans les graphes d'une base de données. Cette méthode voulait prouver la puissance du formalisme des graphes conceptuels dans un contexte particulier. Nous avons dès lors travaillé sur l'amélioration de cette méthode de recherche et sur sa finition. Nous allons maintenant en exposer brièvement le principe. Comme nous l'avons précédemment vu dans ce chapitre, il existe des règles de dérivation, de spécialisation et de généralisation inhérentes aux graphes conceptuels. Cette méthode utilise ces règles. Elle fonctionne simplement par l'introduction d'un graphe dont un concept est une inconnue.

Grâce au formalisme des graphes conceptuels, nous pouvons introduire une méthode de recherche simple et efficace. Le principe de cette recherche se fait simplement par l'introduction de graphes dans la base de données. Cette insertion s'effectue comme pour des connaissances. Il s'agit donc d'introduire un graphe comportant des concepts génériques ou individuels ainsi que des relations avec des inconnues. Les cas suivant illustre nos propos :



SCHEMA 7.26 : Exemples de graphe de recherche.

Dans le premier graphe, on recherche toutes les occurrences du type *Employé*. Dans le deuxième cas, on recherche le type du référent John. Dans le troisième graphe, l'inconnue se retrouve au niveau de la relation, c'est-à-dire que l'on recherche la relation entre l'employé John et la compagnie ONU. Et enfin dans le dernier cas, la réponse fournie sera le référent de la compagnie pour laquelle John travaille.

Nous remarquons aussi qu'il existe deux types de recherche à proprement parler. D'une part une recherche large comme illustrée dans les graphes ci-dessus. La recherche se fait de manière générale et le résultat sera un référent ou un ensemble de référents. D'autre part nous avons un autre type de recherche qui donnera comme résultat une affirmation ou une négation. Chercher à savoir si John travaille pour l'ONU constitue une telle recherche.

Sans rentrer dans les détails, notons que l'algorithme de recherche se déroule de manière séquentielle. En effet, cet algorithme dresse une table de relations vérifiant la question. Ensuite, il fait une recherche dans cette table pour trouver quelle(s) relation(s) possède(nt) des concepts source et destination validant cette question. Enfin, il ne reste plus qu'à trouver les référents de ces concepts et de cette relation et à rendre accessible la réponse.

Conclusion de chapitre

Le formalisme des graphes conceptuels est très récent. Celui-ci se compose de notions de base qui sont les relations et les concepts. Il dispose aussi d'une hiérarchie des typez délimitée par le type universel et le type absurde. Nous avons vu aussi qu'il existe différentes opérations qui peuvent être appliquées aux graphes conceptuels. Ces opérations sont des opérations de dérivation, de spécialisation et de généralisation.

Dans sa thèse intitulée : *'Un modèle uniforme pour la modélisation et la métamodélisation d'une mémoire d'entreprise'* [GERBE00-1], le Professeur Gerbé analyse différents formalismes et leur capacité à répondre à des besoins qu'il juge indispensable pour représenter une base de connaissances. Ces besoins sont au nombre de trois : la classification et la connaissance partielle, les relations entre les catégories et/ou les instances ainsi que les relation entre les catégories ou les instances et le métamodèle. Les

formalismes étudiés et repris dans ce travail sont aussi au nombre de trois : le modèle Entité-Association, le formalisme UML et les graphes conceptuels. Il ressort de cette comparaison que le formalisme des graphes conceptuels est le seul à répondre efficacement aux trois besoins.

Comme nous avons pu le constater, de par sa nature récente, le formalisme des graphes conceptuels est encore peu développé et non standardisé. Il est donc nécessaire de définir une syntaxe précise et conventionnelle pour un métamodèle de représentation des connaissances sous forme de graphes. Notons qu'une standardisation est en cours. [SOWA01]

Il est important de constater que l'emploi des graphes conceptuels n'est que théorique. Nous entendons par là que cette théorie bâtie par le Professeur Gerbé peut être soumise à la critique. De même, il faut voir dans l'application de cette théorie une tentative de validation par la pratique qui s'apparente plus à un travail de recherche qu'à une implémentation concrète. En effet, notre travail était plus une validation de la théorie qu'un développement de logiciel. Signalons que cette validation n'est à l'heure actuelle pas terminée.

Le prototype en lui-même fonctionne avec deux tables au lieu de neuf pour l'approche Entité-Association. Comme pour cette dernière, l'insertion des connaissances se fait à l'aide de fichier XML traité par des fichiers XSL. Nous avons vu que nous avions collaboré à développer un outil de recherche de connaissances. Cet outil a permis de montrer la puissance et la flexibilité du formalisme des graphes conceptuels. Notons qu'il n'existe aucun outil de ce genre pour l'approche Entité-Association. L'implémentation d'un tel outil ne poserait pas de problèmes mais serait beaucoup plus rigide que celui développé ici.

Le chapitre suivant rappellera les différents objectifs définis pour le prototype et effectuera une comparaison entre les deux approches. De plus, ce chapitre comparera le prototype avec les différentes méthodes de gestion des connaissances exposées dans le troisième chapitre. Il donnera aussi les avantages et les faiblesses relevés du prototype et donnera quelques évolutions possibles permettant de pallier à ces différentes faiblesses.

Huitième chapitre

Evaluation du prototype Anthémis

I. Mise en situation

1. Rappel des objectifs du prototype Anthémis
2. Comparaison modèle Entité-Association et graphes conceptuels

    2.1. Insertion des connaissances
    2.2. Diffusion des connaissances
    2.3. Méthode de recherche
    2.4. Adéquation aux besoins des systèmes de gestion d'entreprise
    2.5. Pré-requis à l'utilisation du prototype

II. Evaluation intrinsèque du prototype Anthémis

1. Le prototype et les différentes méthodes

    1.1. Méthode et outil logiciel
    1.2. Les acteurs
    1.3. Contrôle syntaxique et sémantique

2. Avantages du prototype Anthémis

    2.1. L'insertion et la gestion de la base de données
    2.2. La présentation uniforme
    2.3. Le mécanisme d'auto-régulation
    2.4. Les graphes conceptuels

3. Faiblesses du prototype Anthémis

III. Evolutions du prototype Anthémis

1. Evolutions pour combler les faiblesses
2. Evolutions pour améliorer l'existant

Introduction de chapitre

Dans ce dernier chapitre, nous allons évaluer le prototype Anthémis. Nous en dégagerons ainsi ses objectifs, ses avantages, ses faiblesses ainsi que les évolutions possibles le concernant. Nous réaliserons aussi une comparaison entre les deux approches suivies par le prototype ainsi qu'une comparaison entre celui-ci et les différentes méthodes vues au deuxième chapitre de ce présent travail.

Nous allons donc, dans une première partie, reprendre les trois objectifs du prototype Anthémis et ensuite, nous comparerons les deux approches pour atteindre ces objectifs. Ces deux approches sont, nous le rappelons, l'approche Entité-Association vue dans le sixième chapitre et l'approche graphes conceptuels décrite dans le septième chapitre.

Dans la deuxième partie, nous évaluerons intrinsèquement le prototype Anthémis. Pour ce faire, nous réaliserons une comparaison thématique entre notre prototype et les différentes méthodes vues au deuxième chapitre de ce présent travail. Nous dégagerons également les avantages ainsi que les faiblesses de celui-ci. Par faiblesses, nous entendons les points non abordés durant notre stage par manque de temps mais qui seront, du moins nous l'espérons, pris en compte dans le futur.

Dans la dernière partie, nous détaillerons les évolutions à entreprendre sur notre prototype. Nous parlerons ainsi des évolutions qui permettront de combler ses faiblesses et nous parlerons également des évolutions qui amélioreraient ce qui a déjà été créé.

I. Mise en situation

Le prototype Anthémis a pour but final de devenir un logiciel utilisable par toutes organisations pour gérer les diverses connaissances de celles-ci.

1. Rappel des objectifs du prototype Anthémis

Le premier objectif est de garantir à l'utilisateur la possibilité de pouvoir insérer dans la base de données tous les styles de connaissances. Qu'entendons nous par la possibilité d'insérer tous les styles de connaissances ? En réalité, l'utilisateur devra avant tout insérer dans la base un modèle qui définira la structure que prendrons ses connaissances et ensuite il pourra insérer les connaissances liées à ce modèle. Ainsi, la base sera entre autres capable de gérer aussi bien des connaissances définissant un processus de travail que des connaissances expliquant la hiérarchie de pouvoir se trouvant dans l'organisation.

Le deuxième objectif du prototype Anthémis consiste à garantir aux utilisateurs de pouvoir gérer leurs connaissances insérées dans la bases de données. La gestion des connaissances doit permettre à l'utilisateur de réaliser plusieurs manipulations sur celles-ci. La première manipulation consiste à créer de nouvelles connaissances dans un modèle défini, la seconde à modifier des connaissances et la dernière à supprimer des connaissances.

Le troisième objectif est d'assurer à l'utilisateur une visualisation de ses connaissances ou celles des autres employés selon des pages uniformes. Par uniforme, nous voulons dire que les pages doivent suivre un certain standard et être structurées. Ainsi toutes les connaissances seront représentées par des pages HTML ayant toutes la même structure et permettant une navigation vers d'autres pages liées à la connaissance visualisée.

En conclusion, le logiciel Anthémis devra être utilisable par l'ensemble des employés de l'organisation. Ainsi, tous les employés pourront utiliser le logiciel soit pour

insérer des nouvelles connaissances dans la base, soit modifier des connaissances déjà existantes, soit supprimer des connaissances devenues obsolètes soit encore visualiser leurs connaissances ou celles des autres employés.

2. Comparaison modèle Entité-Association et graphes conceptuels

Nous avons vu dans les chapitres précédents qu'il existait deux approches pour formaliser nos connaissances. Une approche suivant les modèles Entité-Association et une suivant les graphes conceptuels. Nous allons maintenant comparer ces deux approches qui ont été, nous le rappelons, exposées dans le sixième et septième chapitre de ce présent travail.

Avant de commencer cette comparaison, il nous semble important de rappeler la raison pour laquelle nous avons travaillé sur deux approches différentes. Tout simplement car l'approche concernant les modèles Entité-Association est une approche beaucoup plus classique suivant un but plus commercial. En effet, les modèles Entité-Association sont de nos jours maîtrisés, nous en connaissons les limites et les possibilités. Le Professeur Gerbé pouvait donc, suivant l'approche Entité-Association, imaginer construire une base de connaissances et espérer pouvoir assez rapidement commercialiser le logiciel.. L'approche concernant les graphes conceptuels est, quant à elle, beaucoup plus académique, beaucoup plus expérimentale. Elle est expérimentale dans le sens où elle permet d'analyser la puissance comparée des deux formalismes. En effet, nous sommes conscients des capacités et des possibilités qu'offrent les graphes conceptuels mais il faut encore s'assurer de la compatibilité de cette approche avec la gestion d'une base de connaissances.

Comparons maintenant ces deux approches suivant plusieurs thèmes.

2.1. Insertion des connaissances

En ce qui concerne l'insertion dans la base de données, les deux approches ne diffèrent pas. En effet, elles utilisent toutes les deux une base de données relationnelle et garantissent l'insertion des modèles et des connaissances par l'utilisation de fichiers XML.

Par contre en ce qui concerne la structure de cette base de données, une grande différence apparaît. En effet, la modélisation des données est bien entendu différente d'un formalisme à l'autre (l'approche Entité-Association travaillant avec des entités et des associations, l'approche graphes conceptuels travaillant avec des graphes) et cette différence de modélisation entraîne une structure différente dans les tables de la base de données. L'approche Entité-Association utilise neuf tables pour pouvoir représenter les deux niveaux décrits dans le sixième chapitre de ce présent travail (à savoir le niveau modèle et le niveau connaissance). L'approche graphes conceptuels, quant à elle, n'en a besoin que de deux pour représenter l'ensemble des informations nécessaires pour gérer les graphes. En effet, une table contient les concepts et une autre les relations. Une troisième table viendra se greffer pour contenir les différents graphes ou différents modèles de l'utilisateur.

2.2. Diffusion des connaissances

En ce qui concerne la diffusion des connaissances, la diffusion de celles-ci s'effectuent de la même manière pour les deux approches. Effectivement, pour ces deux approches, les différentes connaissances sont accessibles via un Intranet ou l'Internet. les différentes pages visualisées sont quasi identiques que ce soit pour l'approche Entité-Association ou pour l'approche graphes conceptuels. L'agencement de ces pages est identique mais le contenu de celles-ci peut présenter des différences suivant les fonctionnalités proposées. En effet, l'approche graphes conceptuels permet par exemple de lister les concepts de bases se trouvant dans la base de données alors que l'approche Entité-Association ne permet pas cette fonctionnalité car cette approche ne contient pas de métamodèle et donc ne contient pas de concepts de base. Des captures d'écrans de ces différentes pages se trouvent dans le troisième chapitre du présent travail.

2.3. Méthode de recherche

Une grande différence entre les deux approches se retrouve dans la présence d'un outil de recherche avec le formalisme des graphes conceptuels. En effet, l'approche graphes conceptuels permet d'introduire un graphe comportant des inconnues soit dans les concepts génériques, soit dans les concepts individuels, soit encore dans les relations. Le

prototype s'arrange alors pour retrouver l'élément manquant. Cette méthode de recherche n'existe pas du côté de l'approche Entité-Association. Une méthode de recherche pourrait être envisagée mais elle ne serait pas aussi flexible que celle de l'approche des graphes conceptuels. Ceci étant dû à la structuration très stricte et structurée des modèles Entité-Association.

2.4. Adéquation aux besoins des systèmes de gestion de mémoire d'entreprise

Une autre différence qui existe entre les deux approches se retrouve dans leur capacité à répondre aux besoins requis par les systèmes de gestion de mémoire d'entreprise (SGME). En effet, selon le Professeur Gerbé, il existe trois besoins fondamentaux exigés par ces SGME. Ces besoins sont les suivants : la connaissance partielle et la classification, les relations entre les catégories et/ou les instances ainsi que le besoin de pouvoir voir les catégories comme des instances du métamodèle. Pour une analyse plus détaillée de ces besoins, nous vous reportons à la deuxième section du septième chapitre. Signalons que selon le Professeur Gerbé, aucun des formalismes étudiés dans le septième chapitre ne peut répondre efficacement à ces besoins si ce n'est celui des graphes conceptuels. Les modèles Entité-Association s'avéreraient dès lors moins puissants et moins bien adaptés pour créer un système de gestion de mémoire d'entreprise.

2.5. Pré-requis à l'utilisation du prototype

En ce qui concerne les pré-requis d'utilisation concernant les deux approches nous retrouvons d'une part la capacité de traduire son modèle ou ses graphes en langage XML et d'autre part la capacité de modéliser ses connaissances soit en modèle Entité-Association soit en graphes conceptuels. Nous avons déjà vu que le premier pré-requis devra disparaître dans le futur du prototype Anthémis. Alors que pour le second, il est certain qu'il est plus aisé à l'heure actuelle de modéliser selon le formalisme des modèles Entité-Association tout simplement car ces modèles sont répandus et déjà utilisés tandis que les graphes conceptuels sont plus innovants ce qui pourrait amener une contrainte d'utilisation pour l'utilisateur inexpérimenté.

II. Evaluation intrinsèque du prototype Anthémis

Nous allons dans cette partie comparer notre prototype avec les différentes méthodes vues au deuxième chapitre du présent travail. Une fois cette comparaison réalisée, nous dégagerons précisément les avantages ainsi que les faiblesses de notre prototype. Par faiblesse, nous entendons les points non encore pris en compte par le prototype et nous espérons que ceux-ci seront traités dans le futur.

1. Le prototype et les différentes méthodes

Nous allons maintenant comparer notre prototype et les méthodes étudiées dans le deuxième chapitre. Nous vous rappelons qu'il n'existe pas une méthode spécifique de gestion des connaissances. Nous avons étudié dans le deuxième chapitre les méthodes les plus abordées par la littérature. Ces méthodes sont MACAO, KADS, REX, MKSM. Cette comparaison se fera selon trois optiques. Premièrement, une comparaison entre les méthodes et les outils logiciels. Deuxièmement, nous verrons quels sont les acteurs impliqués dans les différentes méthodes ainsi que dans le prototype. Troisièmement, nous verrons ce qui est effectué pour le contrôle sémantique et syntaxique.

1.1. Méthode et outil logiciel

MACAO (Méthode d'Acquisition des Connaissances Assistée par Ordinateur) permet de structurer les connaissances de façon plus ou moins formelle et de les représenter. L'utilisation du langage MONA permet ensuite de les formaliser et le langage LISA de les coder. Nous n'avons donc pas un logiciel à proprement parler et il serait envisageable de recueillir les connaissances avec cette méthode et d'utiliser le prototype Anthémis pour les gérer.

KADS (Knowledge Analysis et Design System) permet dans sa deuxième phase d'élaborer un système de connaissances. Mais les outils logiciels proposés sont lourds et nécessitent un apprentissage particulier. La plus grande différence avec cette méthode

réside donc dans le fait qu'elle exige une expertise importante alors qu'un des buts à long terme du prototype Anthémis est de n'exiger aucun pré-requis pour l'utilisateur.

REX (Retour d'Expérience) fournit lui aussi une méthode de capitalisation des connaissances et un outil logiciel. Concentrons nous sur l'outil logiciel car le prototype Anthémis, rappelons le, n'offre pas de méthode spécifique. Une première différence que l'on peut noter entre les deux logiciels est que REX fonctionne avec une base de données orientée objet tandis que le prototype utilise une base de données relationnelle. Au point de vue de la diffusion, on peut pointer une différence mineure, REX fonctionne dans un Intranet alors que le prototype Anthémis peut, lui, être accessible via Internet. Ensuite, la consultation des connaissances par REX se fait via une recherche de mots clés, alors que pour le prototype la consultation se fait par navigation. Nous pourrions aussi envisager à long terme une consultation des connaissances assistée par un outil de recherche selon la méthode des graphes conceptuels. Enfin, on peut noter deux ressemblances majeures : premièrement, l'utilisation du côté client d'un simple navigateur et deuxièmement, la présence de doublons ou de points de vue différents. En effet, pour l'instant, rien n'est prévu pour gérer la présence de connaissances de même nature dans notre prototype. Ainsi, deux employés peuvent très bien dans deux modèles différents insérer les mêmes connaissances. Mais l'on pourrait bien imaginer de gérer cela dans le futur, soit automatiquement, soit humainement. La présentation des connaissances par un mécanisme de fiches divisées en trois sections dans REX ressemble également au mécanisme de présentation prévu pour le prototype.

Enfin, la quatrième méthode, MKSM (Method for Knowledge System Management) n'offre pas de support logiciel spécifique. Mais l'utilisation du prototype n'est pas compatible avec cette méthode car celle-ci s'appuie sur des outils comme Word, Excel,... La ressemblance que l'on peut relever est la volonté d'être simple d'utilisation et de n'exiger que peu de connaissances dans le chef de l'utilisateur.

Derrière notre prototype, nous ne retrouvons pas réellement de méthodes de capitalisation des connaissances. Par contre, nous pouvons tout de même en ressortir une manière de faire pour cette capitalisation des connaissances. En effet, l'utilisateur avant de travailler sur le prototype pour insérer de nouvelles connaissances devra procéder en trois phases. La première, qui n'est pas une obligation, consiste en une modélisation de ses

connaissances en modèle Entité-Association ou en graphes conceptuels suivant l'approche dans laquelle l'on se trouve. La deuxième consiste en la traduction du modèle ou des graphes en fichier XML. Et la dernière consiste en la traduction des connaissances liées à ce modèle ou aux graphes dans un second fichier XML.

1.2. Les acteurs

Rappelons que le prototype Anthémis ne propose pas de méthode particulière pour le recueil des connaissances. Nous allons donc analyser les différents acteurs au point de vue des supports logiciels proposés par les différentes méthodes et les comparer avec les utilisateurs du prototype Anthémis. Il faut noter que notre prototype est auto-régulateur, c'est-à-dire que ce sont les utilisateurs eux-mêmes qui insèrent les connaissances et leurs modèles et ce sont eux également qui recherchent les connaissances désirées par le biais d'un navigateur.

Premièrement, MACAO divise les acteurs en deux groupes : les novices et les experts. Les experts sont les personnes qui détiennent les connaissances ; les novices, celles qui veulent connaître les connaissances. On remarque que la méthode MACAO propose deux langages, MONA et LISA, pour formaliser les connaissances. Ces deux langages impliquent pour les novices d'être familiarisés avec ceux-ci. Cela constitue donc une contrainte et nécessite l'apprentissage de ces langages spécifiques par les novices contrairement au prototype Anthémis qui n'exigera aucun pré-requis nécessaire à long terme. Les experts, quant à eux, ne font que répondre à des interviews et se font observer. Ces deux acteurs ont des relations durant la période entière de la méthode.

Deuxièmement, KADS divise également les acteurs en deux groupes qui sont le novice et l'expert. Le principe d'interaction est le même à part que pour KADS, il existe deux phases. La première dans laquelle l'expert est en contact avec le novice pour la récolte de connaissances et la seconde où le novice, seul, formalise les connaissances. Mais cette méthode propose des outils logiciels lourds. Les novices élaborant le système de connaissances devront donc disposer de l'expertise spécifique pour l'implémenter.

Troisièmement, la méthode REX suppose des acteurs ayant des connaissances spécifiques pour remplir la base de données tandis que les utilisateurs n'ont aucun pré-requis nécessaire car la consultation des connaissances se fait à l'aide d'un simple navigateur. A ce niveau, le profil utilisateur s'apparente à celui du prototype Anthémis à la différence que l'insertion des connaissances dans la base de données se fait sans aucun pré-requis spécifique lourd pour notre prototype.

Quatrièmement, les acteurs types de la méthode MKSM s'apparentent plus à ceux du prototype Anthémis. En effet, la méthode a été développée dans le souci de ne demander aucune connaissance spécifique à l'utilisateur. Comme pour le prototype, la constitution de la base de données se fait simplement sans utiliser des langages de programmation lourds. La différence reste toujours dans le fait que MKSM ne propose pas un outil logiciel spécifique mais l'emploi de Word, Excel, Visio,...

1.3. Contrôle syntaxique et sémantique

Comme nous l'avons vu, le prototype ne dispose pour l'instant que d'un contrôle syntaxique et non sémantique. Nous entendons par là que la validation de la connaissance se fait par rapport à la structure du modèle inséré. Mais il n'existe aucun système qui permet de vérifier que cette connaissance est juste. Etant au stade du prototype, nous ne savons pas si cela sera pris en charge dans la version définitive du logiciel, mais il nous paraîtrait judicieux d'instaurer un contrôle matériel ou humain.

Au point de vue des différentes méthodes, on constate que MACAO dispose d'un contrôle sémantique car chaque formalisation de la connaissance par le novice est soumise à la validation de l'expert, et ce, au contraire de la méthode KADS dans laquelle la tâche de conceptualisation est complètement séparée de la phase de recueil ce qui entraîne que les connaissances formalisées ne sont pas soumises à l'expert. Il n'existe donc pas de contrôle sémantique dans KADS. De même la méthode REX n'offre pas de contrôle sémantique particulier. De plus, les auteurs refusent de supprimer les doublons. On peut donc retrouver un même processus conceptualisé différemment selon son auteur. Il existe donc dans la base de données des connaissances qui peuvent être contradictoires. Enfin, pour la méthode MKSM, il n'existe pas de contrôle sémantique à l'insertion des données

mais les connaissances sont recueillies auprès de l'expert. Nous supposons donc, comme pour les autres méthodes et pour le prototype Anthémis, que ce que dit l'expert est valide.

Revenons au prototype. Nous avons vu qu'il serait judicieux d'instaurer un contrôle sémantique dans la version final. On pourrait envisager qu'à chaque insertion les connaissances soient vérifiées par une autre personne. Une deuxième solution consiste à utiliser la méthode MACAO. En effet, le logiciel ne proposant pas de méthode spécifique, le recueil de la connaissance à insérer dans le prototype peut se faire selon l'une ou l'autre méthode. Nous suggérons MACAO car cette méthode inclut une validation de l'expert avant l'insertion dans la base de données.

2. Avantages du prototype Anthémis

Nous allons dans cette section parler des avantages majeurs que le prototype Anthémis offre aux utilisateurs.

2.1. L'insertion et la gestion de la base de données

Le premier avantage qui est l'élément innovateur par rapport aux autres logiciels est de permettre à l'utilisateur avant d'insérer ses connaissances de créer son propre modèle pour ses connaissances. Cela assure ainsi une liberté très large dans l'éventail de connaissances que celui-ci veut insérer dans la base de données. En effet, l'utilisateur ne sera plus limité au modèle prédéfini par le créateur de la base de données mais pourra en quelque sorte créer sa propre base de connaissances dans la base de données centralisée. Et ainsi, avoir la possibilité de travailler sur tous les types de connaissances pour lesquelles il aurait créé au préalable un modèle. Ce principe de laisser à l'utilisateur la possibilité de créer ses propres modèles est réalisable grâce à la structure de notre base de données mais aussi grâce à l'emploi des fichiers XML. Effectivement, le langage XML permet aux utilisateurs d'écrire leurs modèles et leurs connaissances suivant un langage simple par rapport au langage SQL. Nous avons vu que les fichiers XML doivent suivre des balises pré-écrites par nos soins, balises implicites pour l'utilisateur. Une fois ces fichiers insérés,

c'est le logiciel qui s'occupe de la traduction des fichiers en requêtes SQL d'insertion dans la base de données.

2.2. La présentation uniforme

Le deuxième avantage est de garantir à l'utilisateur une présentation de ses connaissances suivant des pages HTML uniformes, c'est-à-dire, standardisées et structurées. Effectivement comme vu dans le troisième chapitre, la présentation suivra toujours la même structure pour les pages de présentation. La page sera toujours divisée en trois sections reprenant dans chaque section les éléments importants de la connaissance visualisée. De plus, il sera permis une navigation entre les pages grâce à des liens hypertextes qui pointeront vers d'autres pages en relation avec la connaissance visualisée.

2.3. Le mécanisme d'auto-régulation

Le troisième avantage qui se rapporte au premier est la liberté offerte à l'utilisateur de créer ses propres modèles. C'est donc, un principe d'auto-régulation de l'utilisateur par rapport au logiciel vu que celui-ci le gère tout seul. Il n'a besoin de personne pour créer, supprimer, modifier ou visualiser des connaissances.

2.4. Les graphes conceptuels

Le dernier avantage que nous pouvons dégager se retrouve dans le principe de réaliser notre prototype selon deux approches. En effet, le but ultime du logiciel est de tourner avec le formalisme des graphes conceptuels qui est plus puissant que le modèle Entité-Association. Donc le logiciel, à condition que les graphes conceptuels fassent leurs preuves pratiquement pour la gestion des bases de connaissances, sera à l'avenir plus puissant que les autres logiciels travaillant avec le formalisme classique des modèles Entité-Association. Et de toute façon, même si ceux-ci ne faisaient par leurs preuves, il reste tout de même l'approche Entité-Association qui est innovateur par le biais de sa base de données et sa présentation uniforme.

3. Faiblesses du prototype Anthémis

Dans cette section, nous allons présenter les différentes faiblesses du prototype. Nous sommes conscients que ces faiblesses doivent disparaître et nous proposerons donc dans la section suivante quelques évolutions à entreprendre pour contrecarrer celles-ci. Certaines de ces évolutions ont déjà été entreprises, d'autres ont été pensées mais pas encore implémentées.

La première faiblesse à dégager est celle concernant la maîtrise soit des modèles Entité-Association soit des graphes conceptuels. En effet cette maîtrise peut être atteinte assez rapidement pour des personnes qui ont l'habitude de travailler avec des modèles de données mais pour l'utilisateur inexpérimenté, cette maîtrise peut être assez lourde.

Une seconde faiblesse pourrait se retrouver dans le contrôle sémantique des connaissances. Effectivement, nous avons vu que le prototype garantissait un contrôle syntaxique des connaissances en vérifiant que les connaissances étaient correctes par rapport aux modèles insérés préalablement. Par contre aucune vérification de la sémantique des connaissances n'a été créée. Ne faudrait-il donc pas associer au prototype un administrateur système ? Nous en reparlerons dans la section suivante.

Il n'existe pas pour le moment de méthode de sécurité ou de priorité des connaissances insérées. En effet, il est possible pour l'instant à n'importe quel utilisateur de modifier les connaissances des autres, voir même de supprimer celles-ci.

Pour finir, il serait intéressant d'avoir la possibilité pour les utilisateurs d'accéder à une bibliothèque de modèles déjà créés pour leur éviter ainsi de devoir toujours créer de nouveaux modèles pour leurs connaissances.

III. Evolutions du prototype Anthémis

Dans cette partie, nous envisagerons des évolutions à entreprendre sur le prototype Anthémis afin de répondre aux faiblesses que nous venons de voir et ensuite des évolutions à entreprendre afin d'améliorer ce qui a déjà été créé.

1. Evolutions pour combler les faiblesses

Premièrement, il serait important que l'utilisateur ne doive plus traduire son modèle Entité-Association ou ses graphes conceptuels en fichier XML. Il faudrait créer un programme permettant la traduction automatique d'un modèle Entité-Association ou des graphes conceptuels en langage XML. Comme signalé dans le troisième chapitre, cette tâche est réalisable pour les modèles Entité-Association car réalisé par le programme DB-Main. En ce qui concerne les graphes conceptuels, il existe une méthode développée par un étudiant nous précédant sur le projet mais celle-ci n'a pas été validée.

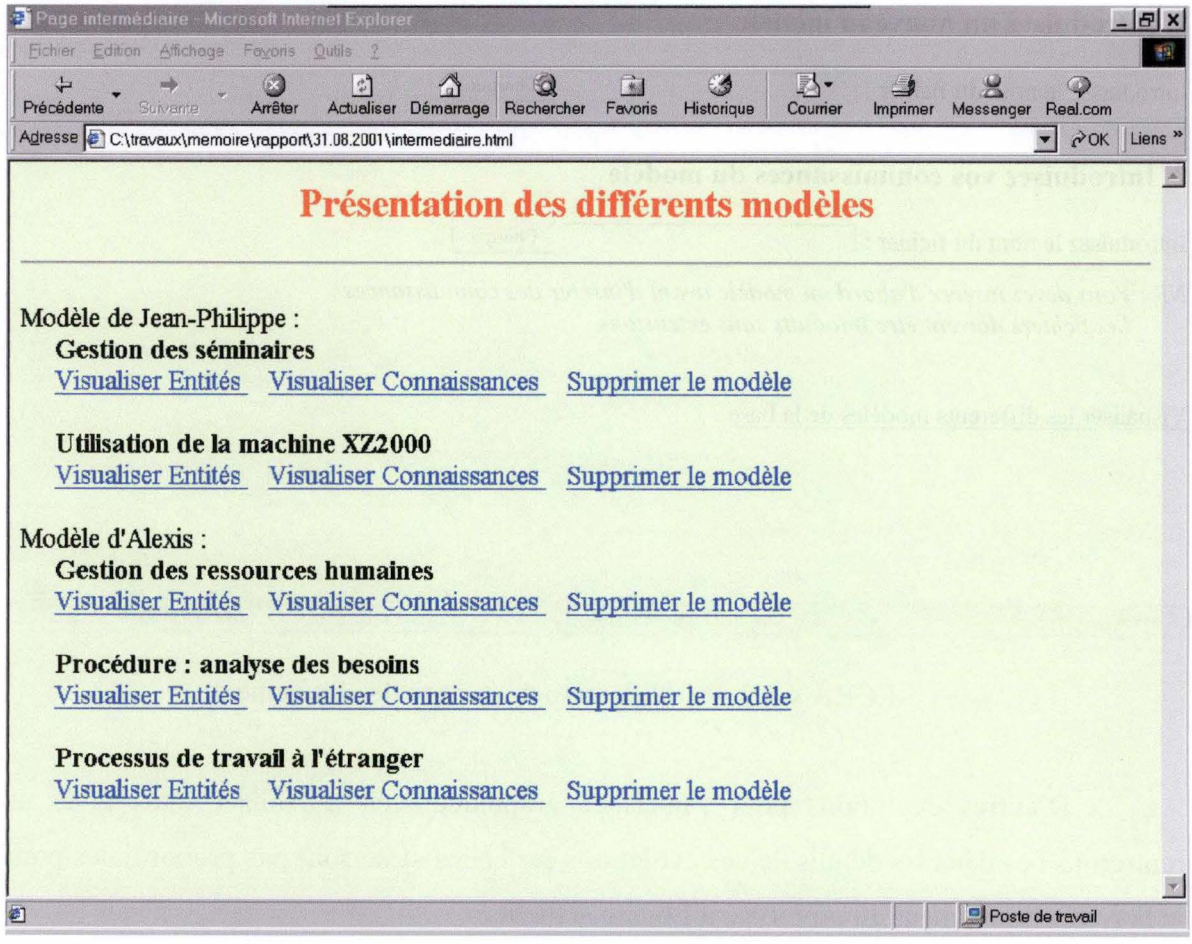
Deuxièmement, en ce qui concerne le contrôle sémantique, une solution apportée pourrait être la création d'un nouveau poste dans l'organisation tel que l'administrateur système. Ce poste aurait comme rôle de contrôler les connaissances insérées dans la base de connaissances et de vérifier si certaines connaissances ne sont pas contradictoires afin d'en avertir les créateurs. Ce poste ne viendrait donc pas à l'encontre du principe d'auto-régulation offert aux utilisateurs mais ne servirait simplement qu'au contrôle sémantique des différentes connaissances.

Troisièmement, il nous semble primordial d'intégrer au prototype des fonctions de partage, de sécurité d'accès des connaissances. Il doit être permis uniquement au créateur d'une connaissance de modifier ou de supprimer ses connaissances ou même de créer de nouvelles connaissances dans son propre modèle. Les autres utilisateurs n'auraient qu'un accès en lecture aux connaissances des autres employés de l'organisation.

Quatrièmement, en ce qui concerne la bibliothèque des modèles, cette fonctionnalité existe déjà en partie sur le prototype concernant l'approche Entité-Association. En effet, dans cette approche, il a été créé une table TX_PACKAGE permettant d'insérer dans cette table les entités concernant les types d'un modèle. Il faudrait donc aller plus loin dans l'idée en permettant de sauvegarder dans une bibliothèque les modèles déjà créés, dans leur totalité. De même pour l'approche selon le formalisme des graphes conceptuels, une évolution a été entreprise concernant la création d'une troisième table. Cette troisième table contiendrait les différents graphes représentant les modèles de données et s'occuperait de leur gestion.

2. Evolutions pour améliorer l'existant

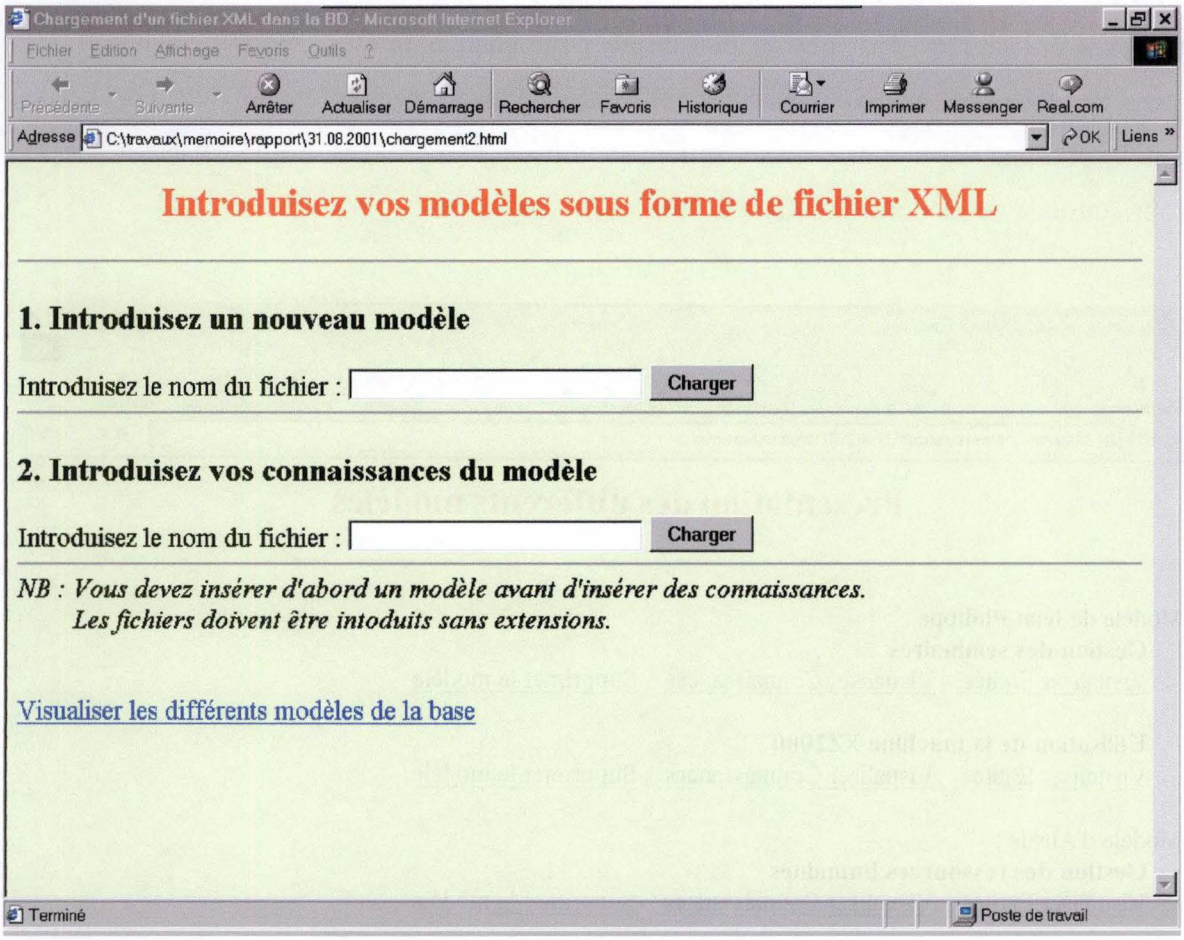
Une amélioration concernerait évidemment l'interface graphique qui est fortement sommaire pour l'instant. Il faudrait entre autres pour le modèle Entité-Association, une fois la gestion de différents modèles pris en compte, utiliser une page intermédiaire entre la page principale (ECRAN 3.1) et la page reprenant les entités concernant les connaissances (ECRAN 3.4). Cette page intermédiaire reprendrait l'ensemble des modèles des différents employés. Sur cette page, nous retrouverons ainsi, le label de tous les modèles créés par les différents utilisateurs ainsi que la possibilité pour chaque modèle de soit visualiser les entités de ce modèle, soit visualiser les entités des connaissances ou même de supprimer le modèle et donc les connaissances qui y sont liées dans le cas où l'utilisateur est le propriétaire de ce modèle. (ECRAN 8.1)



ECRAN 8.1 : Page intermédiaire de présentation des modèles.

Pour réaliser cette page quelques petites modifications sont nécessaires. Tout d'abord, nous devons ajouter une balise dans le fichier XML concernant le modèle qui

serait la balise <auteur>. Cette balise doit se trouver juste après la balise <libelle> référençant la balise <model>. Ensuite, nous devons modifier la table TX_MODEL de la base de données en lui ajoutant l'attribut auteur. De même, nous pourrions envisager d'améliorer la page de chargement du prototype (ECRAN 3.1) en réalisant d'autres liens (ECRAN 8.2). Cette page de chargement permettrait toujours d'insérer un nouveau modèle et les connaissances correspondant à ce modèle. Mais, nous aurions sur cette page un seul lien qui permettrait d'accéder à la page intermédiaire décrite dans l'ECRAN 8.1.



ECRAN 8.2 : Modification de la page de chargement.

D'autres évolutions pour l'interface graphique sont à réaliser mais nous ne rentrerons pas dans les détails de ces évolutions car celles-ci ne sont pas primordiales pour le bon fonctionnement du prototype à l'heure actuelle.

Il ne faut pas non plus oublier de commencer à travailler sur une gestion de plusieurs modèles, même si la structure de la base de données permet cette gestion de plusieurs modèles. Il faut répercuter les changements dans le code déjà implémenté. Et une

fois cette gestion de plusieurs modèles réalisée, se posera la question de savoir si l'on permet dans la visualisation des connaissances d'un modèle, des liens vers des connaissances d'autres modèles ainsi que la question des doublons. Les doublons sont des connaissances qui se retrouvent dans plusieurs modèles différents.

Conclusion de chapitre

Nous avons donc dans ce dernier chapitre repris les deux objectifs généraux du prototype Anthémis et nous avons comparé l'approche Entité-Association avec l'approche graphes conceptuels, toutes deux suivies pour la réalisation du prototype Anthémis. Nous pouvons donc dire que l'approche graphes conceptuels est le but ultime du prototype vu ses capacités et ses possibilités plus grandes que l'approche Entité-Association.

Nous avons également comparé notre prototype avec les différentes méthodes abordées dans le deuxième chapitre du présent travail et de cette comparaison, nous en avons dégagé les avantages et les faiblesses de notre prototype. Les avantages du prototype se retrouvent essentiellement dans la capacité de gérer un large éventail de connaissances suivant des modèles différents ainsi que dans la présentation des connaissances de manière standardisée et structurée. Les faiblesses, quant à elles, sont des points non encore pris en considération vu l'état d'avancement de notre prototype mais qui seront certainement pris en compte dans l'avenir de celui-ci.

En réponse à ces faiblesses nous avons par après présenté des évolutions possibles pour contrecarrer ces faiblesses. Ces évolutions se présentent comme des pistes de recherche afin d'aboutir à une solution implémentable pour les combler. Nous avons également proposé des évolutions pour améliorer le prototype existant. Ces améliorations se retrouvent essentiellement au niveau de l'interface graphique qui est à l'heure actuelle très sommaire car celle-ci a servi essentiellement pour s'assurer du bon fonctionnement des fonctions qui tournaient derrière les différentes pages.

Conclusion

La gestion des connaissances est une pratique en pleine expansion. Les organisations d'aujourd'hui ont compris l'importance de ce domaine. Mais la littérature n'est pas unanime sur la définition de la gestion des connaissances. En effet ce domaine reste difficile à définir et est très abstrait. Nous avons retenu le fait que pour gérer les connaissances, il faut récolter les informations, les interpréter et bien sur diffuser les connaissances ainsi engendrées. Nous avons vu qu'il est important à l'heure actuelle de prendre en compte cette ressource et de la gérer adéquatement. Bon nombre d'organisations passent d'une structure de connaissances individuelles à une structure de connaissances collectives. Les conséquences en jeu sont importantes notamment au niveau concurrentiel. Enfin, nous avons vu qu'un des moyens pour gérer cette connaissance est l'informatique. Celle-ci permet de stocker, de gérer et de diffuser un grand nombre de connaissances.

En matière de gestion des connaissances, plusieurs méthodes et outils existent. Les particularités et l'efficacité diffèrent d'une méthode à l'autre. En effet, certaines méthodes sont dites empiriques et d'autres construites. Les premières se basent essentiellement sur l'observation d'un expert sur son lieu de travail tandis que les secondes présentent la particularité d'être plus structurées. Parmi celles-ci, on retrouve la méthode MACAO (Méthode d'acquisition des connaissances assistée par ordinateur), KADS (Knowledge Analysis and Design System), REX (Retour d'Expérience) et MKSM (Method for Knowledge System Management). Nous avons ainsi présenté le fonctionnement de ces différentes méthodes et vu que ces méthodes ne sont pas des outils de gestion des connaissances même si certaines sont complétées par des supports logiciels.

En ce qui concerne la pratique, nous avons travaillé sur un prototype permettant de gérer les connaissances. L'idée originale de ce prototype réside dans la différence entre les deux approches abordées pour sa création. La première approche se base sur le modèle Entité-Association et la deuxième approche sur le formalisme des graphes conceptuels. Le but ultime du prototype est de fournir un outil de gestion des connaissances abordable par tous les individus d'une organisation en suivant le formalisme des graphes conceptuels. Le logiciel développé ne demandera à long terme aucun pré-requis spécifique de la part de

l'utilisateur. La plus grande innovation réside dans le fait que cet utilisateur pourra introduire son propre modèle de données et, par après, ses propres connaissances. Ainsi l'utilisateur ne sera pas limité à un modèle unique prédéfini mais pourra créer ses propres modèles. Le prototype permet aussi la diffusion et le partage des connaissances de manière standard et uniforme.

Pour le développement du prototype Anthémis, l'emploi d'un certain nombre de technologies était nécessaire. Du côté serveur, nous avons dû employer une base de données ainsi qu'un protocole de communication avec celle-ci. Le choix effectué par le Professeur Gerbé est le système de base de données ORACLE et le protocole de communication ODBC. Ces choix se justifient par le fait que ces technologies sont largement répandues. Pour la diffusion des connaissances, nous avons employé un serveur WEB local ainsi que la technologie ASP. Le serveur WEB local choisi par le Professeur Gerbé est Personnal Web Server de Microsoft. Notons que l'incertitude demeure toujours quant au choix définitif du serveur WEB. En effet, les deux plus courants, Apache et Internet Information Server, présentent tous les deux des particularités différentes ainsi que leurs avantages et leurs inconvénients. Enfin, du côté client, le prototype nécessite un simple navigateur pour son utilisation. Le choix de ce navigateur reste la propriété de l'utilisateur, libre à lui de choisir parmi ceux offerts sur le marché.

En ce qui concerne les langages de programmation utilisés, le prototype en utilise plusieurs, bien répandus à l'heure actuelle. Premièrement le langage HTML qui est un langage de balises permettant la représentation de pages WEB. Deuxièmement, le langage XML qui est un dérivé du premier langage mais qui est beaucoup plus flexible car le programmeur peut définir ses propres balises. Troisièmement, le langage XSL qui permet la mise en forme des documents XML. Quatrièmement, le langage Jscript qui, comme son nom l'indique, est un langage de script qui s'insère dans les pages HTML et les fichiers ASP. Cinquièmement, l'interface ADO qui propose des fonctions génériques qui permettent la communication entre l'interface et la base de données. Ces différents langages ont été choisis par les personnes nous précédant sur le développement du prototype ainsi que par le Professeur Gerbé. Nous avons vu que ces langages étaient bien adaptés à la gestion de base de données et à l'interaction avec le prototype.

La première approche que nous avons étudiée est l'approche selon le modèle Entité-Association. Cette approche se base sur une base de données relationnelle qui contient des tables de données contenant elles-mêmes des lignes et des colonnes. Ce système est le plus largement répandu. Nous avons vu que le prototype utilisait neuf tables réparties en deux niveaux. Le niveau modèle permet de gérer les différents modèles de chaque utilisateur et le niveau connaissance permet de gérer les connaissances contenues dans ces modèles. Cette approche fonctionne en deux temps. La première étape consiste en l'insertion du modèle. Cette insertion s'effectue au moyen d'un fichier XML dont les balises ont été prédéfinies. Notons qu'une DTD permet la validation de ces fichiers XML. La deuxième étape consiste en l'insertion des connaissances. Cette insertion se fait, elle aussi, sous forme de fichier XML. Nous avons également vu que le prototype utilise des fichiers XSL. Ce type de fichier s'applique à un fichier XML et génère un nouveau fichier texte qui permet d'insérer de manière automatique le modèle ou les connaissances dans la base de données. La dernière particularité du prototype est l'utilisation d'une interface graphique. Cette interface permet à l'utilisateur d'accéder aux différentes fonctionnalités du prototype de manière intuitive. Elle se compose d'une succession de pages WEB permettant la diffusion des connaissances au sein de l'organisation. Notons encore que tous les pré-requis nécessaires à l'heure actuelle pour l'utilisateur disparaîtront à long terme, cet objectif étant majeur dans le développement du logiciel.

La deuxième approche se fait selon le formalisme des graphes conceptuels. Ce formalisme est très récent car il a été développé en 1984 par SOWA. Les graphes conceptuels se composent principalement de concepts représentés par des rectangles qui sont reliés par des relations conceptuelles représentées par des ronds. Nous avons vu qu'il existe une hiérarchie des types ainsi que plusieurs opérations qui peuvent être appliquées aux graphes conceptuels et qui permettent leur transformation. Le Professeur Gerbé présente dans sa thèse une analyse de différents formalismes pour représenter les connaissances et les confronte à trois besoins essentiels. Ces besoins sont : la classification et la connaissance partielle, les relations entre catégories et/ou instances ainsi que les relations entre les catégories ou les instances et le métamodèle. Les formalismes étudiés sont le modèle Entité-Association, l'Unified Modeling Langage (UML) et les graphes conceptuels. Nous avons montré que le formalisme des graphes conceptuels est plus apte à répondre à ces besoins. Nous avons présenté aussi un métamodèle développé par le Professeur Gerbé pour représenter les connaissances. Nous en avons expliqué différents

composants qui sont les graphes, les éléments, les concepts et les référents. Nous avons vu ainsi qu'il est nécessaire de compléter ce métamodèle pour disposer d'une représentation uniforme. Enfin, l'utilisation du prototype selon cette approche s'effectue en trois étapes. L'insertion d'une hiérarchie de concepts, d'une hiérarchie de relations et l'insertion des données. Ces insertions s'effectuent comme pour le modèle Entité-Association à l'aide de fichiers XML et XSL. Nous avons également vu qu'il existe un outil de recherche puissant permettant de rechercher parmi la connaissance. L'élaboration du prototype visait à tester l'efficacité du formalisme pour développer un outil permettant de gérer la connaissance. Nous avons vu que cette puissance est vérifiée notamment à travers l'outil de recherche.

Enfin, nous avons mené une comparaison du prototype avec les différentes méthodes d'acquisition de la connaissance ainsi qu'entre les deux approches suivies. Une des grandes différences est que l'approche basée sur le modèle Entité-Association utilise neuf tables, alors que l'approche selon les graphes conceptuels en utilise deux. De plus, la présence d'un outil de recherche montre une différence entre les deux approches. En effet, le prototype avec le formalisme des graphes conceptuels contient un outil de recherche qui a permis de démontrer leur puissance. Enfin, pour l'instant, les pré-requis demandés à l'utilisateur sont plus importants pour les graphes conceptuels car ce formalisme est récent et n'est pas encore très répandu. En ce qui concerne les différentes méthodes de capitalisation des connaissances, elles diffèrent principalement du prototype car celui-ci est avant tout un support logiciel. Au point de vue des différents acteurs amenés à entrer en contact avec le logiciel, nous avons vu que certaines méthodes s'apparentent au prototype car peu ou aucun pré-requis ne sont exigés. Au point de vue du contrôle syntaxique, les méthodes et le prototype se rapprochent car tous effectuent un tel contrôle. Au point de vue du contrôle sémantique, seule la méthode MACAO en propose un. Il serait donc judicieux d'instaurer un tel système dans le prototype. Nous avons vu également quelques avantages du prototype comme la prise en compte d'un grand nombre de types de connaissances ainsi qu'une présentation de la connaissance de manière standard et uniforme. Il est évident que plusieurs évolutions sont possibles et que par exemple il serait intéressant pour l'utilisateur de disposer d'une bibliothèque reprenant les différents modèles compris dans la base de données.

En conclusion, nous pouvons dire que la gestion des connaissances s'avère capitale pour les organisations. Celles-ci doivent disposer de méthodes et d'outils puissants

pour soutenir cette gestion. Le développement du prototype a montré que l'approche Entité-Association répondait bien aux objectifs attendus. Nous avons également montré qu'en utilisant le formalisme des graphes conceptuels pour gérer la connaissance, nous disposions d'un outil plus puissant et plus flexible. Le prototype Anthémis comme nous l'avons indiqué dans le dernier chapitre est toujours dans un état de prototype et doit encore subire plusieurs évolutions. Ainsi la vérification sémantique, les contrôles d'accès aux différentes connaissances, la gestion simultanée de plusieurs modèles, la bibliothèque de modèles sont des éléments à prendre encore en considération.

Bibliographie

1. Références

1.1. Internet

[AUSSENAC1]
Aussenac N. ; Problèmes méthodologiques liés à la construction d'un Modèle Conceptuel avec MACAO ; Page web de l'IRIT « l'Institut de Recherche en Informatique de Toulouse » ; date de création non communiquée ; consultée en juillet 2001 ; adresse : www.irit.fr/ACTIVITES/EQ_SMI/PUBLI/nada-nathJAC94.html

[AUSSENAC2]
Aussenac N. ; Enjeux d'une acquisition des connaissances basée sur l'explication d'un modèle plus générique de l'expertise ; Page web de l'IRIT « l'Institut de Recherche en Informatique de Toulouse » ; date de création non communiquée ; consultée en juillet 2001 ; adresse :www.irit.fr/ACTIVITES/EQ_SMI/PUBLI/jac93.Nathalie.html

[BARTHES98]
Barthès J-P. ; Les systèmes à base de connaissances : la méthode KADS ; Page web d'HEUDIASYC « HEUristiques et DIAgnostiques des SYstèmes Complexes » ; créée le 12 juin 1998 ; consultée en juillet 2001 ; adresse : www.hds.utc.fr/~barthes/IA03/KADS.html

[DESS SID97]
Etudiants du DESS SID (Systèmes Informationnels et Documentaires) ; La gestion des connaissances ; Page WEB de L'université Charles de Gaulle-Lille 3 ; créée le 14 mars 1997 ; consultée en juillet 2001 ; adresse : http://www.univ-lille3.fr/UFR/idist/gid/Mement.HTM

[ERMINE96]
Ermine J-L. ; MKSM, méthode pour la gestion des connaissances ; Page web du programme européen MCX « Modélisation de la CompleXité » ; date de création non communiquée ; consultée en juin 2001 ; adresse : www.mcxapc.org/ateliers/8/mksm.htm

[FILEMAKER01]
FileMaker ; Support ODBC Primer ; Site de FileMaker, logiciel permettant de partager ses informations à travers un réseau ou le WEB ; date de création non communiquée ; consultée en juillet 2001 ; adresse : http://www.filemaker.com/support/odbc_primer1.html

[NETCRAFT01]
Netcraft ; Netcraft Web Server Survey ; Site de Netcraft, société qui offre des services en relation avec le World Wide Web et sa sécurité ; créée en 2001 ; consultée en juillet 2001 ; Adresse : http://www.netcraft.com/survey/

[PILLOU01]
CommentCaMarche.net ; Introduction au modèle relationnelle ; Page WEB provenant de CommentCaMarche.net ; créée en 2001 ; consultée en août 2001 ; adresse : http://www.commentcamarche.net/relation/relintro.php3

[SOWA01]
Sowa J.F., Conceptual graphs, 2001. (http://users.bestweb.net/~sowa/cg/cgstand.htm).

[W3C98]
W3C ; HTML 4.0 Spécification ; Page du World Wide Web Consortium ; créée le 24 avril 1998 ; consultée en juillet 2001 ; adresse : http://www.w3.org/TR/1998/REC-html40-19980424/

[W3C99]
W3C ; XSL Transformation (XSLT) Version 1.0 ; Page du World Wide Web Consortium ; créée le 16 novembre 1999 ; consultée en juillet 2001 ; adresse : http://www.w3c.org/TR/xslt

[W3C00]
W3C ; Extensible Markup Language (XML) 1.0 (Second Edition) ; Page du World Wide Web Consortium ; créée le 6 octobre 2000 ; consultée en juillet 2001 ; adresse : http://www.w3c.org/TR/2000/REC-xml-20001006

1.2. Arcticles

[GERBE96]
Gerbé O., Guay B, Perron M, Using conceptual Graphs for Method Metamodeling, ICCS'96.

[GERBE00-02]
Gerbé O., Mineau G.W, Keller R.K, Conceptual graphs, metamodeling and notation of concepts, ICCS'2000.

[JACOB00]
Jacob R., Gérer les connaissances : Un défi de la nouvelle compétitivité du 21ème siècle, 2000.

[LYMAYEM99]
Limayem M., Frini A., Gestion du savoir dans les organisations, in, Gérer les NTIC, Fiches 22, CEFRIO, 1999.

[SECK98]
Seck M.T, Corenthin A. , Procédure de stockage et d'organisation des données dans une base de données, 1998.

1.3. Ouvrages

[GERBE00-01]
Gerbé O., Un modèle uniforme pour la modélisation et la métamodélisation d'une mémoire d'entreprise, 2000.

[GOYETTE99]
Goyette H., notes du cours : technologies de développement Internet, 2000.

[HAINAUT94]
Hainaut J-L., Bases de données et modèles de calcul. Outils et méthodes pour l'utilisateur, InterEditions, 1994.

[LADD00]
Ladd E., O'Donnel J, Programmation Internet, HTML 4, XML et JAVA 2. Ressources d'experts, Campus Press février 2000.

[NONAKA97]
Nonaka I., Takeuchi H., La connaissance créatrice. La dynamique de l'entreprise apprenante, De boek Université, 1997.

[POLANYI66]
Polanyi M., *The tacit dimension*, éd. Rouledge & Kegan Paul, London, 1966.

[PRAX00]
Prax, J-Y, Le guide du Knowledge Management, 2000, Dunod.

[SOWA84]
Sowa J.F, Conceptual structures : information processing in mind and machine, 1984.

[SVEIBIG00]
Sveibig K. E., *Knowledge Management. La nouvelle richesse des entreprises. Savoir tirer profit des actifs immatériels de sa société*, éd. Maxima, Paris, 2000.

[TARONDEAU98]
Tarondeau J-C, Le management des savoirs, PUF, 1998.

2. Lectures complémentaires

2.1. Internet

Apache group ; About Apache ; site du Apache Group ; date de création non communiquée ; consultation en juillet 2001 ; adresse : http://httpd.apache.org/ABOUT_APACHE.html

Boulogne F. ; PWS : un serveur web avec windows 95/98 ; page personnel de Francis Boulogne ; créée en août 1998 ; consultée en juillet 2001.
Adresse : http://home.nordnet.fr/~frboulogne/pws/pws.htm

Microsoft ; Présentation rapide des tâches de Internet Information Server ; site de Microsoft ; page crée en 1998 ; consultée en juillet 2001.
Adresse : http://www.pem.ch/iishelp/iis/htm/core/iitasks.htm

Rajagopal R. ; Comparing Apache and Internet Information Server ; Site de NetworkComputing ; créée le 4 décembre 2000 ; consultée en juillet 2001 ; adresse : http://www.networkcomputing.com/unixworld/1124/1124uw.html

2.2. Articles

Barthelme-Trapp F., Vincent B., Analyse comparée de méthodes de gestion des connaissances pour une approche managériale, Xième Conférence de l'Association Internationale de Management Stratégique, 13-14-15 juin 2001.

Büyüközkan G., Maire J-L., Capitalisation des connaissances des entreprises pour un benchmarking, septembre 1998.

Gerbé O. , Conceptual graphs for corporate knowledge repositories, ICCS'1997.

Sciences Humaines, La dynamique des savoirs, Hors-Série n°24, Mars-Avril 1999.

2.3. Ouvrages

Donsez D., Programmation procédurale avec SQL, 2000.

Roegel D. , Cours de Oracle : SQL, 1998.

Tisseyre R. C., Knowledge managment, Hermès Science publication, 1999.

Ruggles R., Knowledge managment tool, Butterworth-Heineman, 1997.

Hainaut J-L., Bases de données et modèles de calcul Outils et méthodes pour l'utilisateur Cours et exercices, Dunod, 2000.